Frédéric Boulanger
Daniel Krob · Gérard Morel
Jean-Claude Roussel
Editors

Complex Systems Design & Management

Proceedings
of the Fifth International Conference
on Complex Systems Design
& Management
CSD&M 2014

Springer

CESAMES
Center of Excellence on Systems Architecture, Management, Economy and Strategy

Complex Systems Design & Management

Frédéric Boulanger · Daniel Krob
Gérard Morel · Jean-Claude Roussel
Editors

Complex Systems Design & Management

Proceedings of the Fifth International Conference on Complex Systems Design & Management CSD&M 2014

Springer

Editors
Frédéric Boulanger
Supélec Département Informatique
Gif-sur-Yvette Cedex
France

Daniel Krob
LIX/DIX
Ecole Polytechnique
Palaiseau Cedex
France

Gérard Morel
Université de Lorraine
Vandoeuvre-lès-Nancy
France

Jean-Claude Roussel
Airbus Group Innovations
Toulouse Cedex
France

ISBN 978-3-319-11616-7 ISBN 978-3-319-11617-4 (eBook)
DOI 10.1007/978-3-319-11617-4

Library of Congress Control Number: 2014949169

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Introduction

This volume contains the proceedings of the Fifth International Conference on "Complex System Design & Management" (CSD&M 2014; see the conference website: http://www.csdm2014.csdm.fr for more details).

The CSD&M 2014 conference was jointly organized on November 12–14, 2014 at the Salons de l'Hôtel des Arts et Métiers in Paris (France) by the two following founding partners:

1. The non-profit organization C.E.S.A.M.E.S. (Center of Excellence on Systems Architecture, Management, Economy and Strategy),
2. The Ecole Polytechnique – ENSTA ParisTech – Télécom ParisTech – Dassault Aviation – DCNS – DGA – Thales "Engineering of Complex Systems" chair.

The conference benefited of the permanent support of many academic organizations such as CEA List, Conservatoire National des Arts et Métiers (CNAM), Ecole Centrale de Paris, Ecole Polytechnique, Ecole Supérieure d'Electricité (Supélec), ENSTA ParisTech and Télécom ParisTech which were deeply involved in its organization.

Special thank also goes to Airbus Group, Dassault Aviation, DCNS, Digiteo Labs, Direction Générale de l'Armement (DGA), EDF, Faurecia, Institut de Recherche Technologique (IRT) SystemX, MEGA International, Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche and Thales which were the main industrial and institutional sponsors of the conference. The generous specific support of Airbus Group shall be especially pointed out here.

We are also grateful to many non-profit organizations such as Association Française d'Ingénierie Système (AFIS), International Council on Systems Engineering (INCOSE) and the "Pôle de compétitivité" Systematic Paris-Region which strongly supported our communication effort.

All these institutions also helped us a lot through their constant participation to the organizing committee during the one-year preparation of CSD&M 2014.

Many thanks therefore to all of them.

Why a CSD&M Conference?

Mastering complex systems requires an integrated understanding of industrial practices as well as sophisticated theoretical techniques and tools. This explains the creation of an annual *go-between* forum at European level (which did not existed yet) dedicated both to academic researchers and industrial actors working on complex industrial systems architecture and engineering. Facilitating their *meeting* was actually for us a *sine qua non* condition in order to nurture and develop in Europe the science of systems which is currently emerging.

The purpose of the "Complex Systems Design & Management" (CSD&M) conference is exactly to be such a forum, in order to become, in time, *the* European academic-industrial conference of reference in the field of complex industrial systems architecture and engineering, which is a quite ambitious objective. The last four CSD&M 2010, CSD&M 2011, CSD&M 2012 and CSD&M 2013 conferences – which were held in end of October 2010, December 2011, December 2012 and December 2013 in Paris – were the first steps in this direction. In 2013, there were almost 300 participants who came from 20 different countries with an almost perfect 50/50 balance between academia and industry, which measures the growing success of the CSD&M conference.

Our Core Academic - Industrial Dimension

To make the CSD&M conference this convergence point of the academic and industrial communities in complex industrial systems, we based our organization on a principle of *complete parity* between academics and industrialists (see the conference organization sections in the next pages). This principle was first implemented as follows:

- the Program Committee consisted of 50 % academics and 50 % industrialists,
- the Invited Speakers came in a balanced way from numerous professional environments.

The set of activities of the conference followed the same principle. They indeed consist of a mixture of research seminars and experience sharing, academic articles and industrial presentations, software and training offers presentations, etc. The conference topics cover in the same way the most recent trends in the emerging field of complex systems sciences and practices from an industrial and academic perspective, including the main industrial domains (aeronautic & aerospace, transportation & systems, defense & security, electronics & robotics, energy & environment, health & welfare services, media & communications, software & e-services), scientific and technical topics (systems fundamentals, systems architecture & engineering, systems metrics & quality, systemic tools) and system types (transportation systems, embedded systems, software & information systems, systems of systems, artificial ecosystems).

The 2014 Edition

The CSD&M 2014 edition received 66 submitted papers, out of which the program committee selected 22 regular papers to be published in these proceedings, which corresponds to a 33 % acceptance ratio which is fundamental for us to guarantee the high quality of the presentations. The program committee also selected 33 papers for a collective presentation during the poster workshop of the conference.

Each submission was assigned to at least two program committee members, who carefully reviewed the papers, in many cases with the help of external referees. These reviews were discussed by the program committee during a physical meeting held in C.E.S.A.M.E.S. office in Paris by the May 19, 2014 and via the EasyChair conference management system.

We also chose 12 outstanding speakers with various industrial and scientific expertise who gave a series of invited talks covering all the spectrum of the conference, mainly during the two first days of CSD&M 2014. The first and second day of the conference were especially organized around a common topic - Systems Modeling, Simuling and Decision Aid - that gave coherence to all invited talks. The last day was finally dedicated to a special “thematic session”, followed by presentations of all accepted papers and four parallel events (two system-focused tutorials and two workshops held by IBM).

Furthermore, we had a poster workshop, for encouraging presentation and discussion on interesting but ”not-yet-polished” ideas, and a software tools presentation session, in order to provide to each participant a good vision on the present status of the engineering tools offer.

Acknowledgements

We would like finally to thank all members of the program and organizing committees for their time, effort, and contributions to make CSD&M 2014 a top quality conference. A special thank is addressed to the C.E.S.A.M.E.S. non-profit organization team which managed permanently with an huge efficiency all the administration, logistics and communication of the CSD&M 2014 conference (see http://www.cesames.net/en/).

The organizers of the conference are also greatly grateful to the following sponsors and partners without whom the CSD&M 2014 event would just not exist:

Founding Partners

- Center of Excellence on Systems Architecture, Management, Economy and Strategy (C.E.S.A.M.E.S.),
- Ecole Polytechnique - ENSTA ParisTech - Télécom ParisTech - Dassault Aviation – DCNS - DGA - Thales chair “Engineering of Complex Systems”,

Academic Sponsors

- CEA List,
- Conservatoire National des Arts et Métiers (CNAM),
- Ecole Centrale de Paris,
- Ecole Polytechnique,
- Ecole Supérieure d'Electricité (Supélec),
- ENSTA ParisTech,
- Télécom ParisTech,

Industrial Sponsors

- Airbus Group,
- Dassault Aviation,
- DCNS,
- Direction Générale de l'Armement (DGA),
- EADS,
- EDF,
- Faurecia,
- Mega International,
- Thales,

Institutional Sponsors

- Digiteo labs,
- Institut de Recherche Technologique (IRT) SystemX,
- Ministère de l'Education Nationale, de l'Enseignement Supérieur et de la Recherche,

Supporting Partners

- Association Française d'Ingénierie Système (AFIS),
- International Council on Systems Engineering (INCOSE),
- Pôle de compétitivité Systematic Paris-Region,

Participating Partners

- Atego,
- Dassault Systemes,
- MathWorks,
- MEGA International,
- Obeo,

– PragmaDev,
– SQUORING Technologies,
– The CoSMo Company.

July 2014

Frédéric Boulanger – Ecole Supérieure d'Electricité (Supélec)
Daniel Krob – C.E.S.A.M.E.S. & Ecole Polytechnique
Gérard Morel – Université de Lorraine
Jean-Claude Roussel – Airbus Group Innovations

Conference Organization

Conference Chairs

General Chair

Daniel Krob, president, C.E.S.A.M.E.S. & institute professor, Ecole Polytechnique – France

Organizing Committee Chair

Frédéric Boulanger, professor, Supélec – France

Program Committee Chairs

Gérard Morel, professor, Université de Lorraine – France (academic co-chair)
Jean-Claude Roussel, senior expert systems engineering, Airbus Group Innovations – France (industrial co-chair)

Program Committee

The PC consists of 30 members (15 academic and 15 industrial) of high international visibility. Their spectrum of expertise covers all of the conference topics.

Academic Members

Co-chair

Gérard Morel	Université de Lorraine – France

Other Members

Manfred Broy	Technische Universität München – Germany
Michel-Alexandre Cardin	National University of Singapore – Singapore
Vincent Chapurlat	Ecole des Mines d'Alès – France
Alain Faisandier	Mapsysteme – France
Timothy Ferris	UNISA – Australia
Paulien M. Herder	University of Delft – Netherlands
Claude Laporte	Ecole de technologie supérieure – Canada
Anatoly Levenchuk	TechInvestLab & INCOSE Russia – Russia
Juan Llorens	Technical University of Madrid – Spain
Dominique Mery	Université de Lorraine/LORIA – France
Patrick Millot	LAMIH – France
Yoshiaki Ohkami	Keio University – Japan
Paul Valckenaers	Catholic University of Leuven – Belgium
Jon Wade	Stevens Institute of Technology – USA

Industrial Members

Co-chair

Jean-Claude Roussel, Airbus Group Innovations – France

Other Members

Gérard Auvray	Airbus Defense and Space – France
Jean-Pierre Daniel	Areva – France
Alain Dauron	Renault – France
Rainer Ersch	Siemens – Germany
Gauthier Fanmuy	Dassault Systèmes – France
Pascal Gendre	Airbus – France
Greg Gorman	IBM – USA
Alan Harding	BAE SYSTEMS – Great Britain
David Long	Vitech – USA
Clotilde Marchal	Airbus Group – France
Roland Mazzella	Thales – France
Andrew Pickard	Rolls Royce – Great Britain
Garry Roedler	Lockheed Martin – USA
Robert Swarz	MITRE – USA

Organizing Committee

Chair

Frédéric Boulanger	Supélec – France

Other Members

Marc Aiguier	Ecole Centrale de Paris – France
Anas Alfaris	CCES & MIT – Arabia Saudi
Emmanuel Arbaretier	Airbus Group – France
Karim Azoum	Systematic Paris-Region – France
Eric Bonjour	Université de Lorraine ENSGSI – France
Guy Boy	Florida Institute of Technology – USA
Cihan Dagli	Missouri University of Science and Technology – USA
Pascal Foix	Thales – France
Eric Goubault	CEA List – France
Paul Labrogère	IRT SystemX – France
Garry Roedler	Lockheed Martin – USA
François Stephan	IRT SystemX – France
Nicolas Trèves	CNAM – France
Jon Wade	Stevens Institute of Technology – USA
David Walden	Sysnovation & INCOSE – USA

Invited Speakers

Grand Challenges – Society

Alain Berthoz, professor, Collège de France – France
Eymeric Lefort, director Mission Energy, Grand-Lyon – France
Jeroen Kok, chairman, European Travelers Council – Netherlands

Grand Challenges – Industry

Dominique Luzeaux, vice-director, Ministry of Defense – France
Nguyen Thuy, senior engineer, EDF – France
Simon Bradley FRGS, vice-president, Airbus Group – Great Britain

Scientific State of the Art

Dov Dori, professor, MIT & Technion University – USA/Israel
Marc Shapiro, professor, Université Paris VI – France
Jean-Marc Jézéquel, director, IRISA – France

Methodological State of the Art

Paul Eremenko, director of Advanced Technology & Projects, Google – USA
Jacques Printz, institute professor, CNAM – France
Donna Rhodes, senior scientist and SEAri SSRC director, MIT – USA

Tutorials

Using the INCOSE SE Handbook to Engineer Complex Systems: David Walden, Sysnovation, LLC – USA
Building a Foundation for Continual System Engineering and Self-improvement in Complex Systems: Kirstie Bellman, Chris Landauer and Phyllis Nelson, Topcy House Consulting and California State Polytechnic University - USA

Workshops

MOBILEng, IBM – Israel & USA
SoSAE, IBM – Israel & USA
Systems Modeling, Simuling and Decision Aid – IRT SystemX and Systematic Paris-Region - France

Contents

Regular Papers

1 When Systems Engineering Meets Software Language Engineering 1
Jean-Marc Jézéquel, David Méndez-Acuña, Thomas Degueule, Benoit Combemale, Olivier Barais
1 A Language-Oriented Vision for Systems Engineering 1
2 Challenges for SLE from Language Designers' Point of View . . . 4
2.1 Reuse 4
2.2 Variability Management and Languages Families 7
2.3 Verification and Validation 9
3 Challenges for SLE from the Language Users' Point of View 10
3.1 Language Viewpoint 10
3.2 Language Evolution 11
3.3 Language Integration 11
4 Conclusion and Perspectives 12
References 12

2 Dependency Analysis as a Heat Map for Architecture Standardization 15
Johannes Becker, Mark Gilbert, Armin Förg, Matthias Kreimeyer, Donna H. Rhodes, Markus Lienkamp
1 Initial Situation and Outline of Paper 16
1.1 Challenge 17
1.2 Motivation 17
2 Objective 18
3 Approach 18
3.1 Theoretical Background 19
3.2 Prescriptive: Change Prediction Method 21
3.3 Descriptive: Structural Complexity Management 22

4 Use Case 24
4.1 System Definition 24
4.2 Example 1: Gearbox Integration (Technology Push) 24
4.3 Example 2: New Legal Regulations (Requirement Push) 26
5 Conclusion and Future Work 27
References 28

3 User-Based Solutions for Increasing Level of Service in Bike-Sharing Transportation Systems 31
Juste Raimbault
1 Introduction 32
2 Presentation of the Model 33
3 Results 36
3.1 Implementation and Parametrization 36
3.2 Robustness Assessment, Exploration and Calibration . . . 38
3.3 Investigation of User-Based Strategies 40
4 Discussion 41
4.1 Applicability of the Results 41
4.2 Possible Developments 42
5 Conclusion 43
References 43

4 A New Framework for the Simulation of Offshore Oil Facilities at the System Level 45
Marc Bonnissel, Joris Costes, Jean-Michel Ghidaglia, Philippe Muguerra, Keld Lund Nielsen, Benjamin Poirson, Xavier Riou, Jean-Philippe Saut, Nicolas Vayatis
1 Introduction 46
2 Acausal and Hybrid Modeling 47
2.1 Acausal Modeling 47
2.2 Modelica 48
3 Physics Modeling 49
3.1 Fluid Flow 50
3.2 Thermal Transfer 51
3.3 Hydrate/Wax Formation 51
3.4 Control Systems 52
4 Risk Modeling 52
4.1 Endogenous Factors 53
4.2 Exogenous Factors 54
5 Framework 55
5.1 Software Architecture 55
5.2 Monte Carlo Simulation 56
5.3 Multi-level Simulation 56
6 Conclusion and Perspectives 56
References 57

5 Towards an Extended Interoperability Systemic Approach for Dynamic Manufacturing Networks: Role and Assessment of PLMS tandards 59
Emna Moones, Nicolas Figay, Thomas Vosgien, Lyes Kermad, François Stephan, Abderrahman El Mhamedi, El Mouloudi Dafaoui
1 Introduction 60
1.1 Industrial Context 60
1.2 Research Context and Orientation of the Proposal 61
2 SIP Related Work 62
2.1 Positioning According Interoperability State of the Art 62
2.2 Positioning According Test Beds State of the Art 63
2.3 SIP and System Engineering 64
2.4 Architecture and Principles of SIP Test Bed 65
3 Systemic and Its Limitations for a Global Interoperable PLM Approach 66
4 Illustration through ISA95 Case Study 67
4.1 ISA 95 Standard for Enterprise Control Integration 67
4.2 Modelling and Simulation of a DMN Collaboration Process 68
5 Conclusion and Way Forward 70
References 71

6 Flexible Queries over Engineering Data 73
Yishai A. Feldman
1 Introduction 73
1.1 Technology Background 74
1.2 Use Cases for Queries 75
2 Queries and Rules on Engineering Data 76
2.1 Existence Rules 76
2.2 Consistency Rules 76
2.3 Ontological Consistency Rules 76
2.4 Non-circularity Rules 77
2.5 Domain-Specific Rules 77
2.6 Modeling Rules 78
3 Requirements from Flexible Queries 78
4 A Visual Query Formalism 81
5 Semantics 82
6 Implementation 84
7 Conclusion 85
References 85

7 Leveraging Domain Expertise in Architectural Exploration 87
Henry Broodney, Michael Masin, Evgeny Shindin, Uri Shani, Roy Kalawsky, Demetrios Joannou, Yingchun Tian, Antara Bhatt, Imad Sanduka
1 Introduction 88
2 Engineering Complex Systems with Architecture Patterns 89
3 Architecture Patterns - Example Use Case 91
3.1 Antenna Architecture Patterns 94
4 Integration Framework 95
4.1 Architecture Pattern Representation 98
4.2 Data Supporting Architecture Optimization Process 98
5 Architecture Optimization 99
6 Summary and Future Directions 101
References 102

8 Seven Issues on Distributed Situation Awareness Measurement in Complex Socio-technical Systems 105
Maria Mikela Chatzimichailidou, Angelos Protopapas, Ioannis M. Dokas
1 Introduction 105
2 Previous Work 108
2.1 Types of SA Models 108
2.2 Existing SA Measurement Techniques 109
2.3 A Network-Based Approach for Measuring DSA 111
3 Why It Is Not Worthy to Combine the Existing SA Measurement Techniques 111
4 Issues on DSA Measurement 112
5 Conclusion 115
References 116

9 The Hidden Perils of Addressing Complexity with Formal Process – A Philosophical and Empirical Analysis 119
Paul Nugent, Emilio Collar Jr.
1 Introduction 119
2 Complexity and the Rise of Formal Quality Control Processes 120
3 Formal Process as Technology 122
4 Formal Process and Labor Study Literature 123
5 Research Question 124
6 Research Method 124
7 Analysis 125
7.1 Initial Reactions to Process Introduction 125
7.2 Five Years Later: A Deeper Look at CMMI Process 126
8 Conclusion 128
8.1 Philosophy of Technology 128
8.2 Labor Studies 129
8.3 Future Directions 130

References 130

10 A Formal Foundation of Systems Engineering 133
Dominique Luzeaux
1 Introduction 133
2 What Is Necessary for an Adequate Formalization of Systems Engineering? 134
2.1 Informal Presentation of the Formal Foundations 135
2.2 A Gentle Introduction to Category Theory and Categorical Logic 137
3 Formal Introduction of the Framework 140
3.1 A Category of Systems 140
3.2 Closing the Loop with Categorical Logic and Proof Theory 143
3.3 Extensions of the Proposed Framework 144
4 Concluding Remarks 146
References 146

11 Ontology-Assisted Systems Engineering Process with Focus in the Requirements Engineering Process 149
Anabel Fraga, Juan Llorens, Luis Alonso, José M. Fuentes
1 Introduction 149
2 How Is Nowadays the Requirements Engineering Process 151
3 Ontology-Assisted Requirements Engineering Process in the Systems Engineering Process 154
3.1 Controlled Vocabulary 155
3.2 Thesaurus 155
3.3 Light Ontology 156
3.4 Patterns and Representation Schemas 157
3.5 The Ontology in the Center of the Process 158
3.6 Ontology Tools Available 158
4 Applied In 158
4.1 Authoring 158
4.2 Quality 159
4.3 Reuse 160
5 Conclusions 160
References 161

12 How Can Usage Monitoring Improve Resilience? 163
Jean-René Ruault, Frédéric Vanderhaegen, Christophe Kolski
1 Introduction 163
2 State of the Art 164
2.1 Systems Engineering, Architecture, SysML 164
2.2 Resilience Functions 165
3 Design Pattern Fit to Resilient Systems 167

3.1 Functional Architecture: Monitor System's Usage and Current State . . . 167
3.2 Physical Architecture: Usage Monitoring Components . . . 167
3.3 Impacts of the Usage Monitoring Components Upon Functional and Physical Architectures of Whole System . . . 169
3.4 Impacts of the Usage Monitoring Components Upon User Interface . . . 170
4 Case Study: Railway Accident . . . 171
5 Conclusion . . . 173
References . . . 174

13 Executable Architectures Using Cuckoo Search Optimization Coupled with OPM and CPN-A Module: A New Meta-Architecture Model for FILASoS . . . 175
Siddhartha Agarwal, Renzhong Wang, Cihan H. Dagli
1 Introduction . . . 175
2 Literature Review on Current and Past SoS Projects . . . 177
2.1 Recent Work on FILA-SoS . . . 178
3 Flexible Intelligent & Learning Architecture for System of Systems-FILA-SoS . . . 179
3.1 FILA-SoS Model Overview . . . 180
4 Methodology of Generating Executable Architectures . . . 182
4.1 Multi-objective Optimization . . . 183
4.2 Executable Architectures . . . 185
5 Conclusion . . . 190
References . . . 191

14 User-Centered Design for Emotion. A Case Study in Wellness Products . . . 193
Sergio Gago Masagué, Joaquim Lloveras Macià
1 Introduction . . . 193
2 Research Methodology and Data Collection . . . 194
2.1 Aims of the Study . . . 194
2.2 Case Study . . . 194
2.3 Sensory Attributes . . . 195
2.4 Data Collection . . . 196
2.5 Methodology for Data Analysis . . . 196
3 Weak Design Points Detected . . . 199
3.1 Hydromassage Function . . . 199
3.2 Frame of the SPA . . . 200
3.3 Control Interface . . . 201
3.4 Environmental Awareness . . . 201
4 Generalizing Results to Improve Current and Future Designs . . . 201
4.1 Design Guidelines and Points of Control . . . 202

4.2 Summary of Guidelines 202
4.3 Integrating Guidelines in the Design Process 204
5 Results and Discussion 205
References 206

15 A Fuzzy Approach for Assessing Transportation Infrastructure Security 207
Michelle S. Dojutrek, Samuel Labi, J. Eric Dietz
1 Introduction 207
2 A Review of Past Work 210
3 Methodology 213
3.1 Security Rating 213
3.2 Fuzzy Logic Framework 216
3.3 Rules 218
4 Case Study 219
5 Conclusion and Discussion 222
References 223

16 A Verification Approach from MDE Applied to Model Based Systems Engineering: xeFFBD Dynamic Semantics 225
Blazo Nastov, Vincent Chapurlat, Christophe Dony, François Pfister
1 Introduction 225
2 Towards Execution and Validation of DSMLs 227
3 Application in the Field of SE 228
3.1 Phase 1: Executable Metamodel Definition 229
3.2 Phase 2: Semantics Definition 233
4 Application Discussion and Expected Contributions 235
5 Conclusion and Perspectives 237
References 237

17 How to Boost Product Line Engineering with MBSE – A Case Study of a Rolling Stock Product Line 239
Hugo G. Chalé Góngora, Marco Ferrogalini, Christophe Moreau
1 Introduction 239
2 PLE for the Railway Rolling Stock Sector: Origins, Motivations and Generations 242
3 Defining a Reuse Strategy 243
4 Looking Back, Looking Forward 245
5 Model Based Framework for MBSE and PLE 248
6 Application of a 2^{nd} Generation Reuse Strategy to a Metro Platform 251
7 Conclusions and Way Forward 255
References 255

18 An Overview of Multimodal Transport Design and Challenges Underlined by a Carsharing Case Study 257
Aurélien Carlier, Fabien Tschirhart, Frédéric Da Silva, François Stephan, Olivier Thoni, Alix Munier-Kordon, Manel Abid, Lionel Scremin, Ludovic Couturier
1 Introduction 258
2 Multimodal Transportation Design 258
2.1 Definition 258
2.2 Topology and Design 259
2.3 Some Modelling Approaches 259
3 Challenges 260
3.1 Governance 260
3.2 Demand Prediction 261
3.3 Planning 261
3.4 Business Model 262
3.5 Multimodal Supervision 262
4 Carsharing, a New Transportation Mode at the Crossroads of All Challenges 262
4.1 Presentation, Urban Impact and Challenges 262
4.2 Case-Study: Dimensioning Carsharing Fleet within a Multimodal Transportation System 264
5 Conclusion 266
References 267

19 Timed Symbolic Testing Framework for Executable Models Using High-Level Scenarios 269
Mathilde Arnaud, Boutheina Bannour, Arnaud Cuccuru, Christophe Gaston, Sebastien Gerard, Arnault Lapitre
1 Introduction 269
2 Approach Overview 271
3 Interaction Scenarios and Symbolic Simulation 272
3.1 Sequence Diagrams 273
3.2 Symbolic Execution 274
4 Activity Diagrams and Numeric Simulation 275
4.1 Activity Diagrams 275
4.2 fUML Virtual Machine and Discrete-Event Simulation 276
5 Conformance Testing 277
5.1 Conformance Relation 277
5.2 Testing Process 278
6 Experiments 279
7 Related Works 280
8 Conclusion 281
References 281

20 MoSaRT Framework: A Collaborative Tool for Modeling and Analyzing Embedded Real-Time Systems 283
Yassine Ouhammou, Emmanuel Grolleau, Michaël Richard, Pascal Richard, Frédéric Madiot
1 Introduction 283
2 Background and Related Work 285
3 Objectives of MoSaRT Framework 286
4 MoSaRT Design Language 287
5 MoSaRT Analysis Repository 289
5.1 Instantiation of the MoSaRT Analysis Repository 290
6 MoSaRT Usage Scenarios 291
6.1 The Back-end of the MoSaRT Framework 291
6.2 The Front-end of the MoSaRT Framework 293
7 Conclusion 293
References 294

21 Preliminary Hazard Analysis Generation Integrated with Operational Architecture – Application to Automobile 297
Pierre Mauborgne, Samuel Deniaud, Eric Levrat, Eric Bonjour, Jean-Pierre Micaëlli, Dominique Loise
1 Introduction 298
2 Background on Links between Safety and Systems Engineering 298
3 Conceptual Model of Safe Systems Engineering – Requirement Analysis View 299
3.1 Introduction 299
3.2 Safety Concepts Definition 301
4 Our Proposal Concerning the Integration of PHA and MBSE 302
4.1 Inputs and Outputs of the PHA 302
4.2 Global Approach 303
4.3 PHA Approach Related to Threats 304
5 Application to an Example 304
5.1 Determination of the Scenario 304
5.2 Assessment of the Dysfunctional Scenario (Example Used in ISO 26262) 307
5.3 Proposal of Risk Coverage 307
5.4 Result of the PHA 307
6 Conclusion and Outlook 308
References 308

Posters Workshop

Taming the Complexity of Big Data Multi-cloud Applications with Models 311
Marcos Aurélio Almeida da Silva, Andrey Sadovykh, Alessandra Bagnato, Etienne Brosse

Scalability in System Design and Management, the MONDO Approach in an Industrial Project 313
Alessandra Bagnato, Etienne Brosse, Marcos Aurélio Almeida da Silva, Andrey Sadovykh

Flexible Model-Based Simulation as a System's Design Driver 315
Jean-Philippe Schneider, Eric Senn, Joël Champeau, Loïc Lagadec

Putting Real Production Software in the Loop, Methodologies Enabling SW Co-development between OEMs and Tier 1s 317
David Bailey, Gregory Nice, Guillaume Francois

System Engineering, for a Cognitive Sciences Approach 319
Thomas Peugeot

Requirement Authoring Tools: Towards the Concept of Standard Requirement 321
José M. Fuentes, Anabel Fraga, Juan Llorens, Gonzalo Génova, Luis Alonso

iProd: Intelligent Management of Product Heterogeneous Data in the Product Development Process 323
Massimo D'Auria, Silvia Poles, Roberto d'Ippolito

Implementing ISO 26550:2013 Model Based 325
Andreas Korff

Probabilistic System Summaries for Behavior Architecting 327
Michael Borth

www.UniText.fr Information System Concept for Documentation/Project Management 329
Philippe Jarrin, Luc Beaupère

Correct by Prognosis: Methodology for a Contract-Based Refinement of Evolution Models 331
Christoph Etzien, Tayfun Gezgin

Probabilistic Thinking to Support Early Evaluation of System Quality through Requirement Analysis 333
Mohammad Rajabalinejad, Maarten G. Bonnema

System Engineering on 3DEXPERIENCE Platform – UAS Use Case . . . 335
Frédéric Chauvin, Gauthier Fanmuy

Engineering a Parent-System, Designed to Generate Complex Sub-systems in the Field of Defense Planning 337
Wolfgang Peischel

Turning a Suite of Modeling and Processing Tools into a Production Grade System 339
Pascal Rivière, Olivier Rosec

A Geographic Information System Perspective for Large Scale Engineering Systems Design and Management – A Case Study for Sustainable Desalination Network in Saudi Arabia 341
Salma Aldawood, Abdulaziz Alhassan, Abdelkrim Doufene, Anas Alfaris, Adnan Alsaati, Olivier de Weck

If We Engineered Systems Like We Produce Movies 343
Dominique Luzeaux, Thierry Morlaye, Jean-Luc Wippler

Interoperability between Design, Development, and Deployment of Safety – Critical Embedded Systems: CRYSTAL Project 345
Alexandre Ginisty, Frédérique Vallée, Elie Soubiran, Vidal-delmas Tchapet-Nya

Using Orientor Theory for Coherent Decision Making for Application Landscape Design 347
Alexander W. Schneider, Florian Matthes

Reuse / Variability Management and System Engineering 349
Olivier Renault

Accounting for Uncertainty and Complexity in the Realization of Engineered Systems 351
Warren F. Smith, Jelena Milisavljevic, Maryam Sabeghi, Janet K. Allen, Farrokh Mistree

Complexity: Definition and Reduction Techniques Some Simple Thoughts on Complex Systems 353
Jon Wade, Babak Heydari

Requirements for Single Pilot Operations in Commercial Aviation: A First High-Level Cognitive Function Analysis 355
Guy André Boy

Urban Lifecycle Management: System Architecture Applied to the Conception and Monitoring of Smart Cities 357
Claude Rochet, Florence Pinot de Villechenon

Toward Better Integration of Functional and Dysfunctional Models: Safety Architect 359
Frédérique Vallée, Anne-Catherine Vié, Jonathan Dumont, Nataliya Yakymets, Yupanqui Munoz Julho, Agnès Lanusse

Active Experimentation and Computational Reflection for Design and Testing of Cyber-Physical Systems 361
Kirstie L. Bellman, Phyllis R. Nelson, Christopher Landauer

Virtual and Physical Integration of Autonomous Vehicles for an Automated Humanitarian Mission in EasyChair 363
Pieter J. Mosterman, Justyna Zander, Ascension Vizinho-Coutry

Formal Framework for Ensuring Consistent System and Component Theories in the Design of Small Satellite 365
Jules Chenou, William Edmonson, Albert Esterline, Natasha Neogi

SQUORE, an Actionable Dashboard Based on SE Leading Indicators to Manage System Engineering Effectiveness 367
SQUORING Technologies

Author Index 369

When Systems Engineering Meets Software Language Engineering

Jean-Marc Jézéquel, David Méndez-Acuña, Thomas Degueule,
Benoit Combemale, and Olivier Barais

Abstract. The engineering of systems involves many different stakeholders, each with their own domain of expertise. Hence more and more organizations are adopting Domain Specific Languages (DSLs) to allow domain experts to express solutions directly in terms of relevant domain concepts. This new trend raises new challenges about designing DSLs, evolving a set of DSLs and coordinating the use of multiple DSLs for both DSL designers and DSL users. This paper explores various dimensions of these challenges, and outlines a possible research roadmap for addressing them. The message of this paper is also to claim that if language engineering techniques to design any single (disposable) language are mature, the language engineering community needs to fundamentally change its view on software language design. We need to take the next step and adopt the perspective that a software language is, fundamentally, software too and thus the result of a composition of design decisions. These design decisions should be represented as first-class entities in the software languages workbench and it should be possible, during the language lifecycle, to add, remove and change language design decisions with limited effort to go from continuous design to continuous meta-design.

1 A Language-Oriented Vision for Systems Engineering

The engineering of complex software intensive systems involves many different stakeholders, each with their own domain of expertise. It is particularly

Jean-Marc Jézéquel · David Méndez-Acuña · Thomas Degueule ·
Benoit Combemale · Olivier Barais
IRISA, University of Rennes 1, France

Benoit Combemale
Inria, France

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_1

true in the context of systems engineering in which rather than having everybody working with code/model defined in general-purpose (modeling/programming) languages, more and more organizations are turning to the use of Domain Specific Languages (DSLs). DSLs allow domain experts to express solutions directly in terms of relevant domain concepts, and use generative mechanisms to transform DSL specifications into software artifacts (e.g., code, configuration files or documentation), thus abstracting away from the complexity of the rest of the system and the intricacies of its implementation.

The adoption of DSLs has major consequences on the industrial development processes. This approach, a.k.a. Language-Oriented Programming [19], breakdowns the development process into two complementary stages (see Figure 1): the development, adaptation or evolution by *language designers* of one or several DSLs, each capitalizing the knowledge of a given domain, and the use of such DSLs by *language users* to develop the different system concerns. Each stage has specific objectives and requires special skills. Figure 1 depicts the two interdependent processes that continuously drive each other's. The main objective of the language engineering process is to produce a DSL which tackles a specific concern encountered by engineers in the development of a complex system, together with its tooling. Once an appropriate DSL is made available to systems engineers, it is used to express the solution to this specific concern in the final system. However, by definition, DSLs are bounded to evolve with the domain they abstract. Consequently, systems engineers need to be well aware of end users' expectations in order to report their new requirements to the language designers. A new evolved DSL is then produced by the language designers, which is in turn used by systems engineers and so on and so forth. It is worthwhile to note that, although this is unlikely in large companies, these roles can be alternatively played by the same people in smaller organizations.

As a matter of fact, while DSLs have been found useful for structuring development processes and providing abstractions to stakeholders [10], their ultimate value has been severely limited by their user-understanding ambiguity, the cost of tooling and the tendency to create rigidity, immobility and paralysis (the evolution of such languages is costly and error-prone). The development of software languages is a challenging task also due to the specialized knowledge it requires. A language designer must own not only quite solid modeling skills but also the technical expertise for conducting the definition of specific artifacts such as grammars, metamodels, compilers, and interpreters. *"Software languages are software too"* [7] and, consequently, languages development inherits all the complexity of general software development; concerns such as maintainability, re-usability, evolution, user experience are recurring requirements in the daily work of software language engineers. As a result, there is room for application of software engineering techniques that facilitate the DSL construction process. This fact permitted the emergence of what we know as *Software Language Engineering* that is defined as the application of

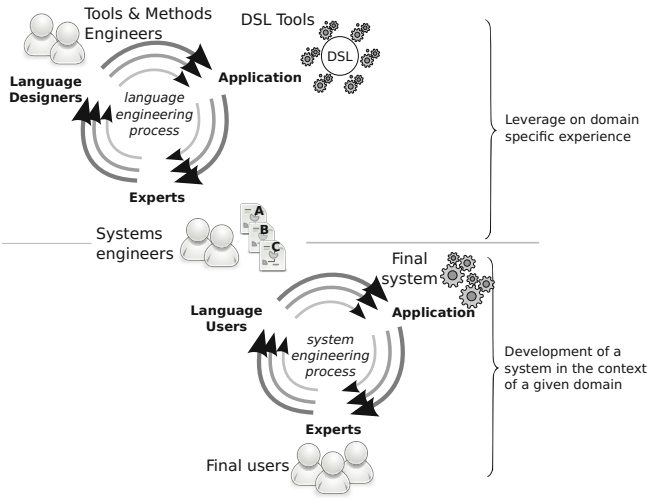


Fig. 1 Language engineering stakeholders

systematic, disciplined, and measurable approaches to the development, use, deployment, and maintenance of software languages [11].

The message of this paper is twofold. First, we claim that language engineering techniques for designing disposable DSLs are close to maturity. However, as we will see, some challenges such as composition, modularity or evolution still need to be addressed. Hopefully, decades of research in software engineering already paved the way and software language engineering should leverage these facilities in order to tackle these challenges. Second, we claim that the common view on software language design should fundamentally evolve. Rather than abstract syntax trees, metamodels, type checkers, parsers, code generators, compilers, etc., we need to model and represent a software language as the composition of a set of language design decisions, concerning, among others, the existing language-units solutions, variation points, features and usage scenarios that are needed to satisfy the requirements. Once we are able to represent software languages, in several phases of the lifecycle, in terms of the aforementioned concepts, changing and evolving software languages will be considerably simplified.

The remainder of this paper is organized as follows. We investigate the underlying challenges of the adoption of DSLs in the development of complex software intensive systems firstly from the points of view of the language designer (Section 2) and secondly from the language user point of view (Section 3). Finally, Section 4 draws some perspectives and concludes.

2 Challenges for SLE from Language Designers' Point of View

From a language designer point of view, the development of new DSLs as well as the evolution of existing ones becomes daily activities. Evolving a DSL usually requires the co-evolution of all its tooling (parsers, textual syntax and graphical syntax editors, compilers, code generators, ...). Besides, language users generally require backward compatibility or tooling for supporting the migration. Consequently, each evolution is costly and error-prone and the software language engineering community still needs to come up with new solutions. To enable this vision that the language design decisions should be represented as first-class entities in the software languages workbench and the it should, during the language lifecycle, be possible to add, remove and change language design decisions against limited effort, his section explores some required software engineering techniques that have been used in the context of software languages engineering for improving i) the reuse, ii) the variability management and iii) the verification and validation. Specifically, we highlight the main challenges that remain to be addressed in each case.

2.1 Reuse

Reusability of software artifacts is a central notion that has been thoroughly studied and used by both academics and industrials since the early days of software construction. Essentially, designing reusable artifacts allows the construction of large systems from smaller parts that have been separately developed and validated, thus reducing the development costs by capitalizing on previous engineering efforts.

It is however still hardly possible for language designers to design typical language artifacts (e.g. language constructs, grammars, editors or compilers) in a reusable way. The current state of the practice most of the time prevents the reusability of language artifacts from one language to another, or from one system to another, consequently hindering the emergence of real engineering techniques around software languages.

Conversely, concepts and mechanisms that enable artifacts reusability abound in the software engineering community. In this section, we present the time-honored concepts of substitutability, inheritance and components, show their relevance for language designers and draw some perspectives and challenges for their inclusion in software language engineering.

Substitutability. In its broadest sense, substitutability is the mechanism that allows the replacement of one software artifact (e.g. code, object, module) with another one under certain conditions. In the context of software language engineering, the considered artifacts (languages, models, abstractions, tools, etc.) are all candidates for substitutability mechanisms, allowing reusing them in different contexts. We propose the notion of types as

interfaces that express the constraints that different artifacts must verify in order to be substituted one another.

The substitution principle has been thoroughly investigated in the object-oriented programming community. It states whether given objects in a program can safely be substituted to other objects based on the subtyping relation that stands between their types [14]. Most object-oriented programming languages feature a mechanism of subtype polymorphism that allows considering the same object through different interfaces (i.e. types), provided they are subtypes one another, enabling facilities such as code reuse or dynamic binding.

In the context of software language engineering, the definition of such types and subtyping relations enables model (i.e. graph of objects) polymorphism, namely the possibility to manipulate a model or program created using a specific language through different tools initially designed for similar yet different languages [16]. Model polymorphism allows tackling a wide range of scenarios that are commonly faced by system engineers. As a concrete example, consider the management of evolution on complex languages such as UML. It is difficult for engineers to deal with this evolution, as all efforts concentrated around a language are lost with subsequent versions; e.g. a transformation defined for UML2.1 cannot be reused for models created using UML2.2 since these are, although semantically close, different languages. Specifying the parameter of such a transformation in terms of model interface allows reusing it for any model that matches this interface: if a subtyping relation can be established between the two versions, model polymorphism allows the reuse of all the tooling across them, even benefiting from dynamic binding for prospective specialization.

However, the currently prevalent modeling frameworks do not provide such type of substitutability mechanisms, and only a few recent research works address them (e.g. [5, 9, 17]). Challenges such as complete language semantics substitutability or concrete syntax replacement still need to be addressed. More generally, using interfaces for specifying the expected features and properties of a language paves the way for language-agnostic, generic manipulation of models and programs. This is particularly relevant in system engineering as engineers need to deal with many different domains and stakeholders, each using his own domain-specific, independently-evolving language.

Extension. The need for language extension arises in many scenarios. DSLs are initially designed and implemented for a restricted set of users with a finite set of features that support their requirements but, most of the time, new requirements will emerge once the language gets effectively used: they tend to grow with the users' needs. Moreover, DSLs are now more and more scattered among different set of users that tackles the same domain, but with their own specificities. In this case, language designers should be able to reuse an existing DSL that contains the basic constructs and features for a particular domain, and extend it with their own business distinctive features.

To support such scenarios where extensions are most of the time unforeseen, DSLs must be designed in a way that facilitates their reuse and extensibility. Conversely, a language designer that extends an existing language should be able to concentrate on its business specificities, seamlessly reusing the base language along with all its tooling. In this regard, real extensibility mechanisms should support the introduction of new constructs, abstractions, or tools without having to understand or recompile the base language source code.

Modularity and Composability. Modularization of software (*i.e.*, components-based software development) is considered an effective mechanism for achieving software reuse and, consequently, reducing costs and time to market. The main principle is to structure software applications as sets of interconnected building blocks that, in turn, are designed in such a way that allows later re-use in other applications. In this context, three of the most important challenges are (1) design and implement components with real potential re-use; (2) design the interfaces that enable crosscutting collaboration among components in a system; and (3) provide components models [12] that offer composition mechanisms for integrating a set of components.

All the aforementioned ideas apply also when the software under construction is a software language; there are benefits in terms of the reduction of construction effort [2]. Nevertheless, those general challenges gain some special connotations analyzed below:

i) Components design (how to breakdown a language?): Decomposing a language in several language modules (a.k.a., language units) is not only about offering a languages benchmark that enables modular definition of metamodels, grammars and semantics; it is just the technical part. The other important challenge is to understand how the language units should be defined so they can be reused in other contexts. What is the correct level of granularity? What are the *"services"* that a language unit should offer for being considered reusable? What is the meaning of a *"service"* in the context of software languages? What is the meaning of a *"services composition"* in the context of software languages?

ii)Languages interfaces (how language units are specified?): The construction of a language unit is not only about implementing a subset of the language but also about specifying its *boundary* (i.e., the set of services it offers to other language units and the set of services it requires from other language units). This fact refers to the classical idea of required and provided interfaces introduced by components-based software engineering approaches. But... what is the meaning of *"provided and required services"* in the context of software languages? We argue that the answer to that question must consider at least two facts: composability and (one more time) substitutability. In the case of composability, required and provided interfaces should provide a mechanism for exposing providing services so they can be consumed

by the required services of other language units. In other words, interfaces are a mechanism for interaction between language units. By the other hand, substitutability refers to the possibility of implementing provided services in different language units so the provided interface becomes also a set of constraints that ensure the safe replacement of the given implementations.

iii) Language units composition (how two language units do interact?): The nature of the interaction between two language units might be different depending on some architectural decisions; extending one language unit with another one is a different situation from a required service of a language unit consuming one or several provided services from other language units. In the case of extension, the base language unit is usually independent of the extensions whereas the extensions have little sense without the base language unit [6]. In the case of required and provided interactions, the requiring language unit usually cannot work without the provided one. We argue that there is a need of composition operators that explicitly define the role of each language unit in a composition.

2.2 Variability Management and Languages Families

One of the main limitations of components-based software development –and of course it is also true in the case of software languages modularization– is the difficulty of designing components that can be actually re-used in other contexts. Despite the design principles and patterns, reusing of software components is not guaranteed. In fact, in many cases the effort of building modularized software is not compensated with the re-usability opportunities.

One of the answers that the software community has found is the idea of variability management. Variability management is a mechanism for explicitly representing the commonalities and differences among a family of software products. A family of products is defined as a set of software applications that have similar purposes and that share some functionality but that is specialized in a particular type of users or situation. The idea is to effectively reuse the implementation of such common functionality and having a repository of "common assets" that implement product features. The process of creating a product by using the family of products is called product derivation. To do so, it is necessary to select the desired product features and to offer a mechanism of composition for integrating the assets corresponding to each feature. This is the main principle of what we know as **"Software Product Line Engineering" (SPLE)** that is a software engineering approach that has demonstrated important benefits in the general case of software development.

As demonstrated in [18], variability management –and the ideas behind SPLC in general– can be applied in the context of software languages for increasing the re-usability and then increasing the productivity of software language engineers. In this context, a family of products actually is a family of

languages where there are some commonalities and some differences (consider as an example the family of OCL variants presented in [20]).

Some of the challenges that should be considered are:

Alignment with the Modularization Approach: It is worth noting that modularization is a prerequisite for addressing variability management. In fact, at the implementation level software modularization and variability management are strongly linked. Each concrete feature expressed in the variability model must correspond to a software component in the architecture so a given configuration can be derived in a concrete functional product. In the case of software languages each feature should be mapped to one (or more) language units that offers the corresponding services. Moreover, in [13] van der Linden *et. al.* present a set of three variability realization techniques at the level of the software modularization schema. Those techniques can be viewed as a set of requirements in terms of modularization and composition of the architecture and they are quite related with the concepts of extension, substitutability and adaptation, some of them discussed in the previous section. How to conjugate all those concepts for effectively define an approach that allows the construction of families of software languages?

Multi-stage Orthogonal Variability Modeling: Typically, a software language specification is intended to define the abstract syntax, the concrete syntax and the semantics of a language. As a result, language units have to contribute to each of those dimensions. In other words, each language unit specification includes a partial definition of the abstract syntax, the concrete syntax, and the semantics. The whole language specification is obtained by putting all the language units together. In [8] the authors observed that there exists some variability between each of those dimensions. Thereby, one language construct (i.e., a concept in the abstract syntax) may be represented in several ways (i.e., several possible concrete syntaxes) and/or may have different meanings (several possible semantics). This analysis remains the same for both the whole language specification and each segment defined in language units. Consequently, we have at least three different dimensions of variability each of them regarding one field of the tuple:

- **Abstract syntax variability or "functional variability":** This variability refers to the capability of selecting the desired language constructs for a particular product as long as the dependencies are respected. Consider for example a family of languages for state machines where concepts such as timed transitions, composite states, or history pseudo-states are optional and are only included if the user of the language needs them. This variability dimension is quite similar to the classical concept of functional variability of SPLC where each feature represents a piece of functionality that may be or not included depending on the specific requirements of a user.
- **Concrete syntax variability or "representation variability":** This variability refers to the capability of offering different representations for

the same concept. Consider for example a language for state machines that can have textual or graphical representations. Note that this type of variability is especially relevant in the context of metamorphics DSLs that we will explain later in this paper.

– **Semantics variability or "interpretation variability":** This variability refers to capability of offering the different interpretations to the same concept. Consider for example the semantics differences that exist between state machines languages explored in [4]. In that work, we can see how, for example, the priorities between conflicting transitions in a state machine are resolved with different criteria. If we are able to manage such variability, the reuse opportunities are drastically increased since we are able to reuse the entire language infrastructure (e.g., editors, abstract syntax trees) for the implementation of different languages that are interpreted according to the needs of specific users.

Note that both representation variability and interpretation variability depend on the functional variability. It makes no sense to select a representation (or interpretation) for a language construct that has not been included as part of the language product. In other words, the configuration of representation and interpretation must be performed only for the construct selected in the functional variability resolution.

2.3 Verification and Validation

Just as any complex software artifact, software languages need to be thoroughly verified and validated. Their complex nature, the different aspects that compose them makes it particularly difficult: is a language really suited for the problems it tries to tackle? Can all programs relevant for a specific domain be expressed in a precise and concise manner? Are all valid programs correctly handled by the interpreter? Does the compiler always generate valid code?

Different techniques have been developed for the V&V of traditional software and are good candidates for adaptation to software languages: among them, we focus on design-by-contract and software testing, and the challenges they need to address for the engineering of software languages.

Design-by-contract [15] advocates the definition of precise interfaces for software components, e.g. using preconditions, postconditions, invariants or types. Contracts can then be checked at different levels to assess the correct interaction of components. In the context of software languages, contracts may be defined on the abstract syntax (e.g. using invariants), on the semantics (e.g. using preconditions and postconditions), etc [17]. Design-by-contract is especially relevant for system engineering as it raises the level of abstraction in which the interaction between the different domains and languages is considered, and makes explicit some of the original requirements on the language. An integrated design-by-contract process for software languages

engineering is expected to bring the same benefits as in traditional software development: precise – structural and behavioral – interfaces, improved error handling, specification-driven definition of artifacts, etc.

Software testing, on the other side, is the most prevalent V&V technique in software engineering. Testing software languages is a challenging activity since all their aspects must be checked: abstract syntax, grammar, semantics, tooling, etc. Furthermore, in this context, test data (i.e. models or programs) are themselves complex artifacts, thus complicating the coverage of representative inputs and the definition of oracles functions [1]. The extensive use of generative programming techniques also raises additional problems due to the gap between generation time and testing time; i.e. the to-be-tested generated artifacts are not known yet when the tests are written. Workarounds on this issue include the automatic generation of test cases together with generated artifacts, which in turns increases the testing activity complexity. Finally, the inherent nature of multi-languages engineering requires not only the different languages to be tested, but also their combination and interaction. Such integration tests should be dedicated to the verification and validation of the composition, reusing the testing effort spent on each of its part.

3 Challenges for SLE from the Language Users' Point of View

From the perspective of the users the emergence of several software languages is also challenging. Despite the overall purpose of constructing DSLs is to facilitate the daily work of systems engineers, dealing with several languages implies not only learning new syntaxes but also interacting with an increasing number of tools: editors, compilers, and code generators among others. The remainder of this section is dedicated to explore some of those challenges that must be addressed to serve this vision that the language design decisions must be represented as first-class entities in the software languages workbench.

3.1 Language Viewpoint

Domain-Specific Languages (DSLs) are plain languages, in the sense that many difficult design decisions must be taken during their development and maintenance, and that they can take different shapes: plain-old to more fluent APIs; internal or embedded DSLs written inside an existing host language; external DSLs with their own syntax and domain-specific tooling. All forms of DSLs have strengths and weaknesses – whether you are a developer or a user of a DSL. The basic trade-offs between internal and external DSLs have already been identified and are subject to extensive discussions and research for several years. A new trend though is observed. DSLs are now so widespread that very different users with separate roles and varied objectives use them. Depending on the kinds of users, roles or objectives, the same form

of DSL (external or internal) might not be the best for everybody. Beyond the unification of the different approaches, it is worthwhile for DSLs to support the ability to be self-adaptable to the most appropriate form (including the corresponding IDE) according to a particular usage or task: we call such a DSL a *metamorphic DSL*.

From the same language description and different interface specifications, we envision the ability to derive various IDEs that can be used accordingly. This vision raises many challenges: systematic methods to evaluate when a form of a DSL meets the expected properties (e.g., learnability); artefact modularization; information sharing, while being able to visualize and manipulate an artefact in a particular representation and in a particular IDE; global mechanism to ensure consistency of the artefacts between these heterogeneous IDEs.

3.2 Language Evolution

By definition, DSLs are bbounded to evolve with the domain they abstract. Consequently, DSLs users need to learn and understand the newly-created abstractions, syntaxes and tools. This raises new challenges in terms of change management and learnability of languages. In order to facilitate the transition, the migration of models from one version of a language to another, aka co-evolution, must be fully supported by the workbench: automatic migration when possible, *diff* computation with explicit user refinement when required, etc. The same situation arises when an older language is replaced with a completely new one, defined independently, but abstracting the same domain.

3.3 Language Integration

The development of modern complex software-intensive systems often involves the use of multiple DSLs that capture different system aspects. In addition, models of the system aspects are seldom manipulated independently of each other. System engineers are thus faced with the difficult task of relating information presented in different models. For example, a system engineer may need to analyze a system property that requires information scattered in models expressed in different DSLs. Current DSL development workbenches provide good support for developing independent DSMLs, but provide little or no support for integrated use of multiple DSLs. The lack of support for explicitly relating concepts expressed in different DSMLs makes it very difficult for developers to reason about information spread across different models.

Supporting coordinated use of DSLs leads to what we call the globalization of modeling languages [3], that is, the use of multiple modeling languages to support coordinated development of diverse aspects of a system. The term "globalization" is used to highlight the desire that DSLs developed in an

independent manner to meet the specific needs of domain experts, should also have an associated framework that regulates interactions needed to support collaboration and work coordination across different system domains.

Globalized DSLs aim to support the following critical aspects of developing complex systems: communication across teams working on different aspects, coordination of work across the teams, and control of the teams to ensure product quality.

4 Conclusion and Perspectives

This paper claims that research conducted in SLE for systems engineering should consider that:

- The first phase of research and development in SLE has matured the technology to a level where industry adoption is wide-spread and few fundamental issues remain for efficiently designing any single (disposable) DSL.
- The traditional view on SLE suffers from a number of key problems that cannot be solved without changing our perspective on the notion of language, and especially of DSL. These problems include i) the lack of first-class representation of design decisions in DSL: since design decisions are cross-cutting and intertwined, they are easy to forget andhard to change, leading to high maintenance costs; ii) the lack of support for explicitly relating different DSLs that makes it very difficult for system engineers to use of multiple DSLs while enabling a coordinated development of the diverse system aspects, and to reason about information spread across artifacts built with different DSLs.
- As a community, we need to take the next step and adopt the perspective that a software language is, fundamentally, software too, that is, the result of a composition of design decisions. These design decisions should be represented as first-class entities in the software language workbench and it should, during the language lifecycle, be possible to add, remove and change language design decisions with limited effort to go from continuous design to continuous meta-design.

Acknowledgments. This work is partially supported by the ANR INS Project GEMOC (ANR-12-INSE-0011), the bilateral collaboration between INRIA and Thales Research & Technology VaryMDE, and the ITEA2 European project MERgE.

References

1. Baudry, B., Ghosh, S., Fleurey, F., France, R., Le Traon, Y., Mottu, J.-M.: Barriers to systematic model transformation testing. Communications of the ACM 53(6), 139–143 (2010)

2. Cleenewerck, T.: Component-based DSL development. In: Pfenning, F., Macko, M. (eds.) GPCE 2003. LNCS, vol. 2830, pp. 245–264. Springer, Heidelberg (2003)
3. Combemale, B., DeAntoni, J., Baudry, B., France, R.B., Jezequel, J.-M., Gray, J.: Globalizing modeling languages. Computer 47(6), 68–71 (2014)
4. Crane, M., Dingel, J.: Uml vs. classical vs. rhapsody statecharts: not all models are created equal. Software & Systems Modeling 6(4), 415–435 (2007)
5. de Lara, J., Guerra, E.: Generic meta-modelling with concepts, templates and mixin layers. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part I. LNCS, vol. 6394, pp. 16–30. Springer, Heidelberg (2010)
6. Erdweg, S., Giarrusso, P.G., Rendel, T.: Language composition untangled. In: Proceedings of the Twelfth Workshop on Language Descriptions, Tools, and Applications, LDTA 2012, p. 7:1–7:8. ACM (2012)
7. Favre, J.-M., Gasevic, D., Lämmel, R., Pek, E.: Empirical language analysis in software linguistics. In: Malloy, B., Staab, S., van den Brand, M. (eds.) SLE 2010. LNCS, vol. 6563, pp. 316–326. Springer, Heidelberg (2011)
8. Grönniger, H., Rumpe, B.: Modeling language variability. In: Calinescu, R., Jackson, E. (eds.) Monterey Workshop 2010. LNCS, vol. 6662, pp. 17–32. Springer, Heidelberg (2011)
9. Guy, C., Combemale, B., Derrien, S., Steel, J.R.H., Jézéquel, J.-M.: On model subtyping. In: Vallecillo, A., Tolvanen, J.-P., Kindler, E., Störrle, H., Kolovos, D. (eds.) ECMFA 2012. LNCS, vol. 7349, pp. 400–415. Springer, Heidelberg (2012)
10. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of mde in industry. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, pp. 471–480. ACM (2011)
11. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels, 1st edn. Addison-Wesley Professional (2008)
12. Lau, K.-K., Wang, Z.: Software component models. Transactions on Software Engineering 33(10), 709–724 (2007)
13. van der Linden, F.J., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer (2007)
14. Liskov, B.H., Wing, J.M.: A behavioral notion of subtyping. ACM Transactions on Programming Languages and Systems (TOPLAS) 16(6), 1811–1841 (1994)
15. Meyer, B.: Applying 'design by contract'. Computer 25(10), 40–51 (1992)
16. Steel, J., Jézéquel, J.-M.: On model typing. Software & Systems Modeling 6(4), 401–413 (2007)
17. Sun, W., Combemale, B., Derrien, S., France, R.B.: Using model types to support contract-aware model substitutability. In: Van Gorp, P., Ritter, T., Rose, L.M. (eds.) ECMFA 2013. LNCS, vol. 7949, pp. 118–133. Springer, Heidelberg (2013)
18. Vacchi, E., Cazzola, W., Pillay, S., Combemale, B.: Variability support in domain-specific language development. In: Erwig, M., Paige, R.F., Van Wyk, E. (eds.) SLE 2013. LNCS, vol. 8225, pp. 76–95. Springer, Heidelberg (2013)
19. Ward, M.P.: Language-oriented programming. Software-Concepts and Tools 15(4), 147–161 (1994)
20. Wende, C., Thieme, N., Zschaler, S.: A role-based approach towards modular language engineering. In: van den Brand, M., Gašević, D., Gray, J. (eds.) SLE 2009. LNCS, vol. 5969, pp. 254–273. Springer, Heidelberg (2010)

Dependency Analysis as a Heat Map for Architecture Standardization

Johannes Becker, Mark Gilbert, Armin Förg, Matthias Kreimeyer, Donna H. Rhodes, and Markus Lienkamp

Abstract. Heavy duty trucks are high variant products with a comparably small production volume per product family. A high degree of specialization regarding utilization scenarios and transportation tasks, as well as strong spreading of functional variability generate increasing numbers of offered variants. The continuous introduction of new legal, technical and customer requirements combined with long product life cycles as well as the need for prolonged technological backward compatibility causes a complexity problem. Architecture standardization is a key lever in reducing complexity by deliberately cutting the number of variants and defining stable interfaces. However, at this point standardization potentials do not seem to be fully exploited.

This paper proposes an architecture standardization method using two approaches complementing product architecture development. First, a prescriptive approach predicts direct and indirect change propagation paths within a generic

Johannes Becker · Mark Gilbert
Technische Universität München, Garching, Germany
e-mail: {j.becker,m.gilbert}@mytum.de

Armin Förg · Markus Lienkamp
Institute of Automotive Technology, Technische Universität München, Garching, Germany
e-mail: {foerg,lienkamp}@ftm.mw.tum.de

Matthias Kreimeyer
MAN Truck & Bus AG, Munich, Germany
e-mail: atthias.kreimeyer@man.eu

Donna H. Rhodes
Systems Engineering Advancements Research Initiative (SEAri),
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: rhodes@mit.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_2

truck architecture, based on component dependencies. Secondly, a descriptive approach identifies geometrical conflicts in the product concept phase and facilitates the introduction of architectural standards, which in turn resolve these conflicts and decouples dependencies within the architecture. Applying these methods serves as a heat map that helps to identify the hot spots for potential standardization in product architectures. It is outlined and illustrated in two examples of change-related conflicts between physical components and product functionality.

Keywords: product architecture, change propagation, dependency analysis, complexity management, architecture standardization.

1 Initial Situation and Outline of Paper

Heavy duty trucks (HDT) are products with a tremendously high number of different operational scenarios and use cases. Their functionality and technical requirements vary greatly depending on their individual purpose (e.g. long-haul logistics, distribution, construction, etc.). The operational demand for a truck is characterized by maximization of payload, reliability, efficiency and uptime. [18, p. 8].

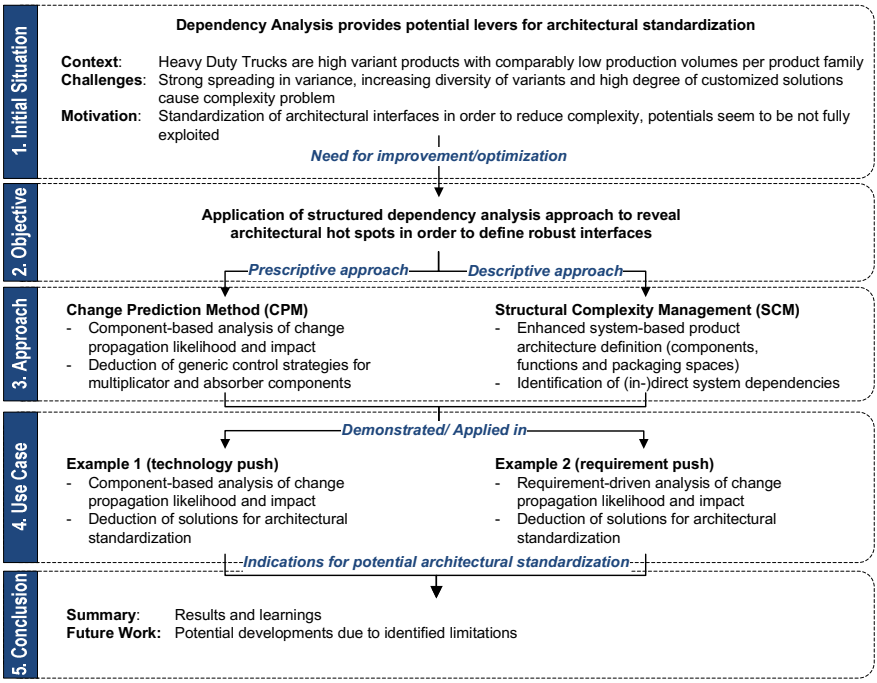


Fig. 1.1 Outline of the paper structure

This paper discusses *product architecture* standardization in HDT using the example of MAN Truck & Bus AG (MAN), a leading German commercial vehicle OEM. MAN's large portfolio of highly configurable vehicles relies on a modular architecture allowing for easy mass customization. The structure of this paper is shown in figure 1.

1.1 Challenge

The *transport solution* needed by a customer is always a combination of the truck and its services and the body or trailer [18, p. 9]. Every single truck is a very individualized product (mass customization [18, p. 6]) serving as a platform for further extensions and modifications by equipment and body manufacturers, i.e. the truck product architecture has to offer adaptability and flexibility beyond the influence of the OEM. MAN's truck product architectures allow functional variants of up to 10^{46} [12, p. 1].

Truck manufacturers cannot benefit from economies of scale due to significantly lower production volumes compared to passenger cars [18, p. 6]. Instead, the focus lies on the modularity and versatility of HDT product architectures.

The high compatibility of components in conjunction with an ever-increasing diversity of variants causes complexity problems. This is due to an increase of functionality provided by new technologies (e.g. hybridization). Additionally, the complexity further rises with increasingly sophisticated national and global legal requirements (e.g. introduction of the EURO VI norm [16, pp. 172-175]). Lastly, HDT product architecture has to remain backwards-compatible and still support systems introduced decades ago, despite the continuous incorporation of new technologies and components (e.g. concurrent use of EURO II/III in developing countries and EURO IV+ in Europe).

The balancing act (to be solved) lies in the generation of new variants based on a structure, which has grown over time and cannot be modified in a radical way in order to retain backwards compatibility. At the same time, it is necessary to ensure that the product architecture is future-proof against upcoming and long term changes in technology and other requirements.

1.2 Motivation

This complexity problem is caused by growing numbers of subsystems or combinations thereof, which have to be incorporated in the product architecture.

Standardization of interfaces as well as geometric boundaries of package configuration were identified to be key levers to mitigate or control this complexity problem. Through standardization potential variant combinations are deliberately excluded from the desired solution space while standardized interfaces facilitate controlling arising change propagation.

However standardization potentials are not fully exploited with regards to HDT product architectures. A considerable amount of manpower is still involved in

clarifying and solving variance issues and further research on modular kits in commercial vehicle design is ongoing [12, p. 2].

Product architecture standardization could resolve this complexity problem and reduce conflict potentials in the early stages of product development.

2 Objective

The objective of this paper is to systematically analyze the *product architecture* of HDT by applying a structured dependency analysis approach to reveal architectural *hot spots* in order to define robust and stable interfaces. Hot spots are understood as the ideal elements to leverage the revealed standardization potentials.

Furthermore, this approach identifies *change propagation* paths as well as deduces means of controlling occurring change propagation. For an unambiguous description of different *packaging space* constellations, a qualitative formalization of packaging spaces is proposed, enabling systematic assignment of components into geometric sections of the physical product structure [12]. Lastly, the objective is to identify interfaces for potential standardization.

The novelty of the paper is constituted by proposing an extended product architecture definition and by applying a combination of two approaches to truck product development.

3 Approach

The approach used in this paper combines prescriptive and descriptive methods in order to identify key elements for architecture standardization. In general, prescriptive methods have the goal to advance the state of the practice using theory-based knowledge while descriptive methods use information and constraints of the state of the practice in order to advance the theory-based state of the art. [20, p. 2]

The *Change Prediction Method* (CPM) [2] allows for prescriptive identification of change-related risks in a product and provides a framework for handling change-critical elements of the system (section 3.2). *Structural Complexity Management (SCM)* [14] allows the definition of extended product architectures from a systems point of view and acts as the descriptive part of our approach.

The interaction between the prescriptive method, product development process, descriptive method and architecture standardization is illustrated in figure 3.1.

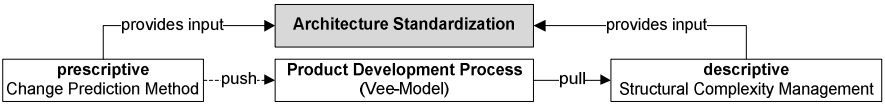


Fig. 3.1 Utilization of prescriptive and descriptive methods for architecture standardization

3.1 Theoretical Background

The classical definition of *product architecture* originates from the early 1990's when the question was raised whether product architectures may be used to simplify product development and to reduce its complexity. This product architecture definition is based on three characteristics: [24, p. 420]

- arrangement of functional elements or functional structure
- mapping of functional elements to physical elements
- specification of the interfaces among interacting physical elements

Product architecture can also be described as a *system*. Definitions of systems have a long history [26, p. 52; 9, p.34]. In this paper, a system is understood as a combination of the definitions in [13, pp. 23-24] and [15, p. 8], considering both system boundaries as well as its inputs and outputs.

An important aspect of product architecture research is to identify means of reducing *complexity*. In systems, complexity is manifested by connectedness, characterized by relationships, and variety, represented by elements. Their diversity and quantity further adds to complexity [19, pp. 22-24]. The complexity of systems can be represented using graph theory or matrix-based approaches [13, pp. 43–61].

Furthermore, complexity can be classified by linking the different dimensions of complexity to strategic technical aspects of a product (see figure 3.2) [25].

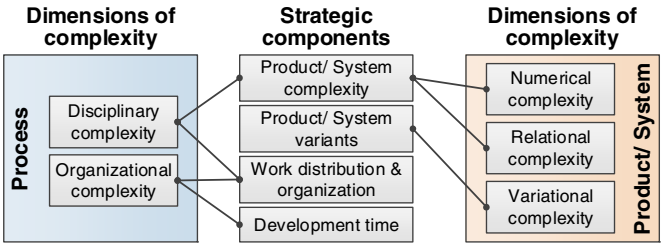


Fig. 3.2 Coherence between dimensions of complexity and strategic components of a system

However, complexity is not an undesired state for a system or a product per se. It comes with opportunities (e.g. capability to control a diversity of variants) and obstacles or negative effects (e.g. numerous changes due to lack of transparency). In competitive market environments it is advantageous to have the ability to cope with complexity. Many prosperous companies work on the edge of manageable complexity, and are successful for exactly that reason [13, p. 20].

Literature discusses two fundamentally different approaches for handling complexity: (1) to avoid and mitigate it, and (2) to manage and control it [13, pp. 31-35].

This paper utilizes matrix-based approaches to model and analyze product architectures as a system. The *Design Structure Matrix (DSM)* is a sort of intra-domain matrix which maps elements of the same nature to each other [6; 11,

p. 2; 22]. A domain contains elements of the same nature [13, p.49].The mapping between DSM elements represents a specific dependency (e.g. physical, spatial, energy, or information) [13, p. 49].

A domain mapped to another domain is known as *Domain Mapping Matrix (DMM)* [3]. DMM was introduced as a complementary form of DSM to overcome its characteristic single-domain limitations. [4, p. 304]

A combination of DSM and DMM complemented with computation logics to derive indirect relationships was introduced as *Multiple-Domain-Matrix (MDM)*. The MDM [17] enables the division of a complex system into subsystems, which are represented by the different domains within the MDM [13, p. 78].

Hence, the *domains* of *components* and *functions* suffice to fully describe product architecture according to the definition mentioned above.

Modularity is one of the most important aspects of product architecture. It defines the way *chunks* of the product architecture are mapped to functions. There are two archetypes of architectures: (1) *modular* and (2) *integral* architectures. [10]

In reality, product architectures are not purely modular or integral but can be classified into different degrees of modularity: (1) *Component Sharing Modularity*, (2) *Component Swapping Modularity*, (3) *Cut to Fit Modularity*, (4) *Bus Modularity*, (5) *Sectional modularity* and (6) *Mix Modularity*. [7, p. 350]

When represented in DSM form, different types of modularity can be visually identified as shown in figure 3.3. *Integral product architectures* (DSM_a) have a very dense DSM. *Bus modular architectures* (DSM_b) have vertical and horizontal lines identifying their bus elements. *Fully integrated product architectures* with serially connected elements have band DSMs (DSM_c). [10, p. 6]

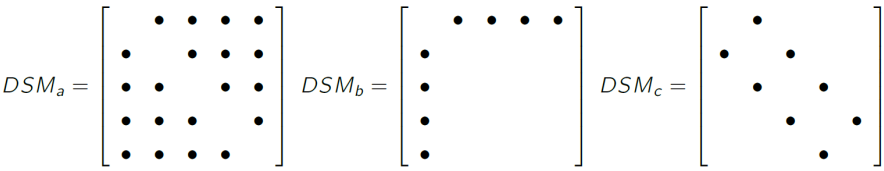


Fig. 3.3 Different types of modularization of product architectures according to [10, p. 6]

Standardization can be used to reduce complexity, costs and lead times in product development. Modular architectures facilitate standardization [24, pp. 431-432].

Component standardization leads to the creation of less component variants. Consequently, these fewer variants are used in higher quantities and benefit from economies of scale and quality improvement by experience.

Architecture standardization (e.g. deliberate elimination of certain possible component variants and component assignments) can be used to mitigate complexity and change propagation.

A formalization of *packaging spaces* is proposed to assign components in early phases of product development into defined sectors of the package. The packaging

space model is of qualitative nature and constructed using simple geometrically adjoining bodies, since only rough information regarding component dimensions and form is available in these early stages. The different vehicle types are generically divided using number and location of axles to derive a parametric typologization model. It is based on cross sections attached to meaningful longitudinal and lateral locations of the vehicle.

Due to its qualitative nature, the model can be universally transferred to other vehicle architectures (e.g. cars). The result is a flexible grid, which is adaptable in terms of distinctness. An emerging packaging space element is modeled as a rectangular prism defined by its six boundary layers. It can be unambiguously visualized using superposed side and top projections of technical drawings (see figure 3.4).

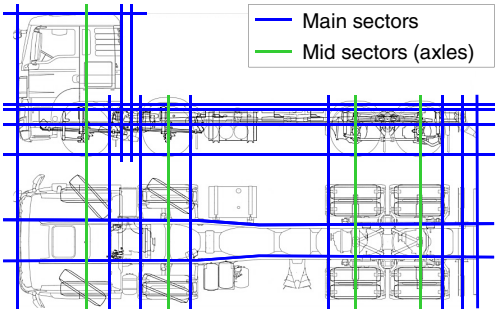


Fig. 3.4 Packaging spaces visualized by superposing qualitative grid and technical drawings in [12, p. 10]

There are other methods for investigating system-to-system interaction (i.e. zonal analysis [1]). However this method is preferably used for aerospace system safety assessment of specific systems, not considering variable system scenarios.

In contrast the proposed model focusses on supporting early decision making by confirming the feasibility to accommodate certain components in emerging packaging spaces.

3.2 Prescriptive: Change Prediction Method

The *Change Prediction Method (CPM)*, initially proposed in 2004 [2], can be used for modular kit development in commercial vehicle design with some adaptations. [12, p. 8-9]

The input used for this method is an innovation planning document mapping product requirements to a generic truck product decomposition. This data is used to generate a DSM describing the change propagation likelihood between elements of the product decomposition. These change propagation dependencies can be modeled in different ways. Firstly, as a *dependency score* where multiple occurrences of dependency pairs from the input document are summed up with

specific weights indicating the level of dependency. Secondly, in a *probabilistic way* where multiple occurrences of dependency pairs in the input document are treated like a binomial distribution of propagation likelihoods.

The probabilistic approach has the advantage of producing bounded values representing propagation likelihoods from 0 to 1. In a first step, these likelihoods describe direct propagation from one component (C_1) to another (C_2). However, this approach also allows the aggregation of indirect propagation paths from component (C_1) to component (C_2) via a path of other components (C_i) [2, p. 792-793]. The combined likelihood matrix can be multiplied with propagation impacts in order to compute a propagation risk matrix.

This component-to-component DSM is used to classify components into different change propagation behaviors by comparing their indegree and outdegree [21, p. 7; 5, p. 13; 23, p. 73-74]. In this classification, components can act as

- *constants*, which are not affected by change. They neither propagate nor absorb changes nor do they add complexity to the change propagation problem.
- *absorbers*, which can absorb more changes than they initiate. Absorbers reduce the complexity of change propagation.
- *carriers*, which propagate a similar amount of change as they absorb. They do not affect the complexity problem.
- *multipliers*, which propagate more change than they absorb. Thereby they amplify changes and increase the complexity of the problem.

Propagation behavior is not an intrinsic feature of a system component. It can be influenced by increasing or decreasing the *contingency margin* of a part [21, p. 13]. Some components, however, might not be changed due to management policy or strategic implications. Typically, these components are bought-in components or involve long development times; they are called *resistors.* Resistors reflect changes [5, p. 14] and usually cause changes in more changeable parts of their surroundings.

In order to increase robustness of product architectures, different measures can be taken regarding change multipliers: They can be isolated and decoupled from other components in order to reduce their probability of receiving and thus multiplying changes, or equipped with sufficient contingency margins mitigating their multiplying behavior in favor of more absorbing behavior [5, p. 14].

Another option is grouping all absorbers and isolating carriers and multipliers by packing them in separated modules [8, p. 7].

3.3 Descriptive: Structural Complexity Management

Structural Complexity Management (SCM) is a generic and standardized approach to tackle engineering problems in product design and development [13, pp. 62-66].

The procedure is divided into five steps, introduced in table 3.1. General starting point for its application is a sound understanding of the actual engineering problem and where its complexity arises from, even before the system definition is approached.

Table 3.1 Procedure of Structural Complexity Management

No.	Step	Activities and operations	Deliverable
1	System definition	A target-aimed system definition is performed by modeling all information within a MDM framework. It involves the definition of a system boundary, an appropriate level of abstraction, identification of domains and the determination of relevant dependencies within the system. The decision about requirement-driven or data based information acquisition is prepared.	MDM Framework
2	Information acquisition	Gathering of native (direct) dependencies between domain elements. To ensure the expressiveness of acquired data, the acquired information must be frequently verified.	Direct system dependencies
3	Deduction of indirect dependencies	To complete the set of required information the different computation schemes are executed to derive the indirect dependencies	Representation of subsets
4	Structure analysis	Graph and matrix-based models are used to carry out a structural analysis of the system. The main objective is to identify meaningful structures of the system and its key elements.	Significant constellations
5	Product design application	Induces learnings from the structural analysis for incremental improvement or redesign of the system as a whole. The reach of incremental improvement is limited while redesign realizes major improvements regarding structure and documented transparency.	Improved system management & design

For the application of the SCM approach choosing a thoughtful abstraction level is important. It defines the depth of detail within the relevant system and has a high impact on data acquisition effort. The trade-off must be made between the level of detail, uncertainty of information, and its acquisition efforts.

An advantageous characteristic of SCM is the feasibility to compute indirect dependencies based on natively acquired direct relationships within systems.

We, for instance, we gained valuable insights regarding potential packaging space conflicts of components due to their assignment to the same packaging space. The likelihood of a packaging space conflict between two components that are actually unrelated is proportional to the strength of the computed indirect dependency. In addition, without knowledge of components assigned to distinct packaging spaces, it is possible to advise their assignment into selected packaging spaces based on their individual connectedness within the system. With this, the structural characteristic of the system can be considered.

4 Use Case

The application of our approach is presented in two separate examples of current product architecture issues.

4.1 System Definition

Our research approach enhanced the 'classical' product architecture definition by considering emerging *packaging spaces*. A vehicle usually offers limited packaging space to accommodate components. Such installation spaces are represented by the packaging spaces domain. The actual system definition and its dependencies between components, functions and packaging spaces are illustrated in figure 4.1.

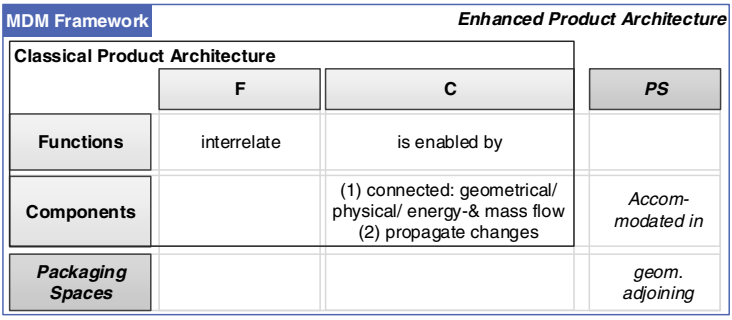


Fig. 4.1 System definition of an enhanced product architecture using functions, components and packaging spaces

4.2 Example 1: Gearbox Integration (Technology Push)

In this case, a new gearbox technology is introduced, which changes size as well as geometrical form of the generic gearbox component. The preexisting gearbox had a wide and flat geometry, whereas the newly introduced gearbox has a narrower but higher design. Thus it exceeds its initial packaging space limits. The new geometry collides with the positioning of the exhaust piping, which previously ran below the gearbox along the truck body frame. As a consequence a change impulse is initiated.

Applying CPM (section 3.2) shows a change in the gearbox has a high likelihood of propagating changes to the exhaust piping. This is visualized in a change propagation matrix (figure 4.2, left). A hot spot in position (1,3) indicates a high propagation likelihood from C_3 (gearbox) to C_1 (exhaust piping). This can be attributed to the fact that gearbox and exhaust piping are very closely spaced, i.e. the geometric contingency margin of the gearbox is very small. This leads to change multiplying behavior in the presence of geometrical modifications of involved components.

Using SCM (section 3.3), the situation analysis is performed in two steps. Initially, the DMM mapping of components to packaging spaces shows no packaging space violation between C_3 and C_1. The geometrical modifications to the gearbox cause a competitive packaging space situation in PS_2 among components C_3 and C_1 (see figure 4.2). Although multiple components sharing a packaging space is not a conflict in itself, it complicates the independent changing of component dimensions amongst others.

From a geometrical point of view, these effects could be remedied by making the cross section of the exhaust piping flat enough for both components to fit next to each other in PS_2. This would, however, negatively influence the vehicle's functionality by reducing the air flow in the exhaust piping. Consequentially, it involves relocating the exhaust piping (as the gearbox is less maneuverable) to avoid collision and a negative impact on performance.

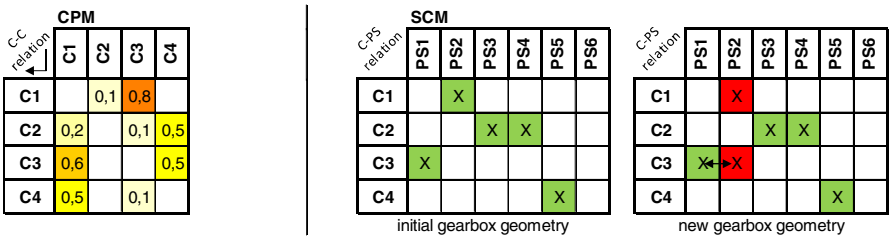


Fig. 4.2 Change Propagation Matrices: left: DSM propagation likelihood, right: DMM component assignment to packaging spaces

Since both CPM and SCM indicate a high likelihood of change propagation between gearbox and exhaust piping, an architectural standard which avoids this interdependency by separating both components from each other is proposed. This could be realized by determining that the exhaust piping always runs from the exhaust silencer at the front right side to the back right side. Introducing this architectural standard, the packaging space conflict is resolved by avoiding any collision with the gearbox regardless of the variant in use. As shown in figure 4.3, components C_1 and C_3 now have significantly lower change propagation likelihood between each other and no longer constitute a hot spot in the propagation likelihood matrix. Moving C_1 to a different packaging space PS_6 (right side of the vehicle) avoids packaging space competition with C_3 regardless of its configuration. This decoupling of the previous component interdependency improves the value robustness [20] of the product architecture by reducing the complexity of future changes to the gearbox.

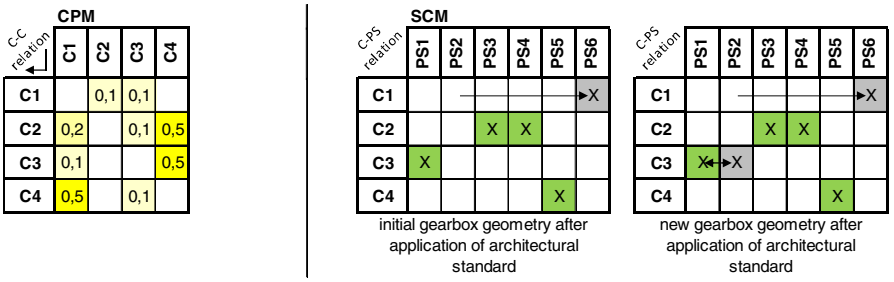


Fig. 4.3 Change Propagation Matrices: left: propagation hot spots eliminated, right: packaging space conflict avoided by relocating components

4.3 Example 2: New Legal Regulations (Requirement Push)

In this example, the event chain of altered legal regulations is discussed. Taking effect from 2014, German legislation requires new vehicles to conform to EURO VI emission limits [18, p. 32]. This causes a change impulse.

The EURO IV/V exhaust gas after-treatment system is complemented by two additional components (AdBlue tank and SCR catalyst). Accommodating both components in the given packaging spaces to fulfill the legal requirements causes a packaging space conflict. Thus, the size, position and form of existing components must be carefully considered to solve this conflict. Backward compatibility and carry over components make this a delicate task as component changes might propagate into the component functionality (e.g. tank size correlates with range) or increase the variance of components (i.e. higher costs).

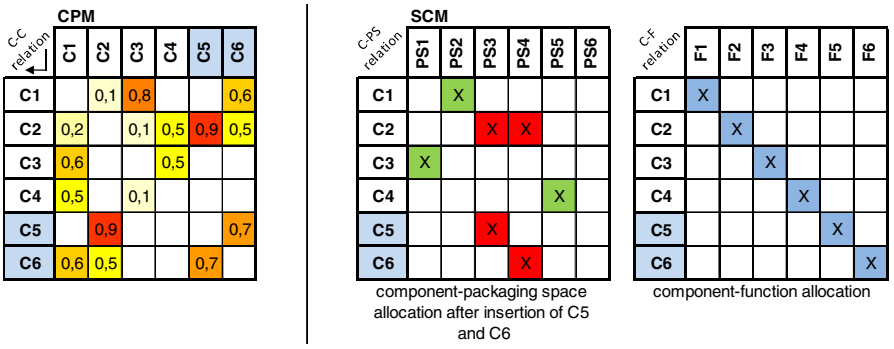


Fig. 4.4 Left: The change propagation likelihood between fuel tank (C_2) and AdBlue tank (C_5) is a hot spot. Right: Both components as well as the SCR system (C_6) compete for volume in packaging spaces PS_3 and PS_4.

Figure 4.4 shows the AdBlue tank (C_5) and SCR system (C_6) in addition to the existing four components. These components have a very high likelihood of propagating change to the fuel tank (left), as they compete for the same packaging

space (middle). For a given wheelbase, packaging space PS_4 only has a limited volume to accommodate all three components. The size of the SCR module is fixed, and the size of the AdBlue tank has to be proportional to the fuel tank (ratio of urea to fuel consumption is 5-7% [16, p. 173]). Thus, given a certain wheelbase, the fuel tank has to be adapted in size to avoid component collisions. Fuel tank volume is not an explicit requirement, though, but rather implied by the need for the greatest range possible. Since the wheelbase limits the total volume, the fuel tank volume is inevitably reduced by the amount necessary to avoid conflicts.

The dimensions of the SCR system and AdBlue tank as well as its connection to the exhaust piping are constant. An architectural standard is defined, which always places the SCR system at the front right side of the truck frame, indicated as PS_4 in figure 4.5. The fuel tank (C_2) is confined to PS_3 and shares this space with the AdBlue tank (C_5). Their respective volumes strongly depend upon each other (see figure 4.5, left) and can be maximized proportionally depending on the available space provided by the given wheelbase. This remaining package space competition and circular change propagation can be handled by always considering fuel tank and AdBlue tank as one coupled system which will be changed altogether if necessary, thereby isolating and internalizing their change multiplying behavior as proposed by [8, p. 4]. The compromise in volume and therefore the maximum range of the vehicle is marked by the asterisks in figure 4.5 (right), indicating altered functionality.

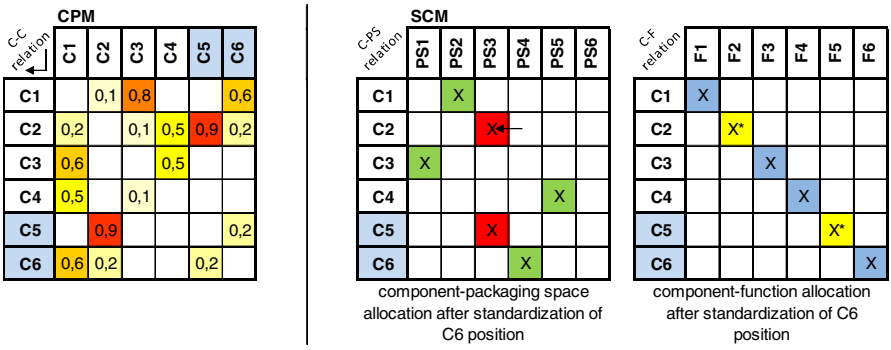


Fig. 4.5 Architecture standardization places the SCR module (C_6) in a dedicated location (PS_4). The fuel tank (C_2) and the AdBlue tank (C_5) compete for volume in PS_3 and have to be modified together due to their high mutual change propagation likelihood (middle). The functionality of C_2 and C_5 is limited by the available volume in PS_3.

5 Conclusion and Future Work

As shown, the proposed approach combines elements of Change Prediction Method and Structural Complexity Management as a decision-making aid for product architecture standardization in the early concept phase of product development.

The CPM can identify areas of high change propagation likelihood and therefore help eliminate *change mines* by confining their influence with by reasonable standardization guidelines. SCM can be used to generically structure and resolve issues of component placement in the truck package.

The proposed approach is limited by the availability of information and its abstraction level, as higher levels of detail require higher data acquisition effort. Furthermore, the approach does not model quantitative aspects like actual component dimensions. Ongoing and further research combines this approach with early digital mock-ups of vehicle concepts automatically generated from requirements specifications in order to generate architectural standards which also take quantitative data into account based on further expansion of the packaging space model [12, p. 7-11].

Acknowledgments. This article contains results of Master's theses from Johannes Becker (Feb to Jul 2013) and Mark Gilbert (Jun to Dec 2012), conducted in a cooperation between Technische Universität München and Massachusetts Institute of Technology. Mr. Becker and Mr. Gilbert would like to thank Dr. Armin Schulz and Dr. Stefan Wenzel of 3DSE Management Consultants for facilitating and supporting their research at MIT. The project was independently funded by the Institute of Automotive Technology at the Technische Universität München, the Systems Engineering Advancements Research Initiative (SEAri) at Massachusetts Institute of Technology, and MAN Truck & Bus AG.

References

[1] Caldwell, R.E., Merdgen, D.B.: Zonal analysis: the final step in system safety assessment [of aircraft]. In: Proceedings of Reliability and Maintainability Symposium, pp. 277–279 (1991)

[2] Clarkson, P.J., Simons, C., Eckert, C.: Predicting Change Propagation in Complex Design. Transactions of ASME 126(5), 788 (2004), doi:10.1115/1.1765117

[3] Danilovic, M., Browning, T.: A Formal Approach for Domain Mapping Matrices (DMM) to Complement Design Structure Matrices (DSM). In: The Sixth Design Structure Matrix (DSM) International Workshop, pp. 1–23 (2004)

[4] Danilovic, M., Browning, T.R.: Managing complex product development projects with design structure matrices and domain mapping matrices. International Journal of Project Management 25(3), 300–314 (2007), doi:10.1016/j.ijproman.2006.11.003

[5] Eckert, C., Clarkson, P.J., Zanker, W.: Change and customisation in complex engineering domains. Research in Engineering Design 15(1), 1–21 (2004), doi:10.1007/s00163-003-0031-7

[6] Eppinger, S.: Model-based Approaches to Managing Concurrent Engineering. Journal of Engineering Design 2(4), 283–290 (1991), doi:10.1080/09544829108901686

[7] Fricke, E., Schulz, A.P.: Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. Syst. Engin. 8(4) (2005), doi:10.1002/sys.20039

[8] Greisel, M., Kissel, M., Spinola, B., et al.: Design for Adaptability in Multi-Variant Product Families. In: Proceedings of ICED 2013. Product, service and systems design, vol. 4. Design Society (2013)

[9] Haberfellner, R., De Weck, O.L., Fricke, E., et al.: Systems Engineering, 12th edn. Grundlagen und Anwendung. Orell Füssli, Zürich (2012)
[10] Hölttä, K., Suh, E.S., De Weck, O.L.: Tradeoff between Modularity and Performance for Engineered Systems and Products. In: 15th International Conference on Engineering Design: Engineering Design and the Global Economy, ICED 2005. Engineers Australia, Barton, A.C.T. (2005)
[11] Kreimeyer, M.: A Product Model to Support PLM-Based Variant Planning and Management. In: Proceedings of the 12th International Design Conference, DESIGN 2012, vol. 3, pp. 1741–1752 (2012)
[12] Kreimeyer, M., Förg, A., Lienkamp, M.: Fostering Modular Kits in an Industrial Brownfield Environment. In: Proceedings of TMCE 2014 (2014)
[13] Lindemann, U.: Methodische Entwicklung technischer Produkte, 3rd edn. Methoden flexibel und situationsgerecht anwenden. Springer, Berlin (2009)
[14] Lindemann, U., Maurer, M., Braun, T.: Structural complexity management. An approach for the field of product design. Springer, Berlin (2009)
[15] Maier, M.W., Rechtin, E.: The art of systems architecting, 3rd edn. CRC Press, Boca Raton (2009)
[16] MAN Nutzfahrzeug Gruppe Grundlagen der Nutzfahrzeugtechnik. Basiswissen Lkw und Bus, Munich (2010)
[17] Maurer, M., Lindemann, U.: Facing Multi-Domain Complexity in Product Development. CiDaD Working Paper Series 3(1), 351–361 (2007), doi:10.1007/978-3-540-69820-3_35
[18] Nielsen, A.: Modularization. MAN Truck & Bus AG Lecture (2014)
[19] Patzak, G.: Systemtechnik. Planung komplexer innovativer Systeme: Grundlagen, Methoden, Techniken. Springer, Berlin (1982)
[20] Ross, A.M., Rhodes, D.H.: Architecting Systems for Value Robustness: Research Motivations and Progress. In: 2nd Annual IEEE International Systems Conference, SysCon 2008, pp. 1–8 (2008)
[21] Shah, N.B., Wilds, J., Viscito, L., et al.: Quantifying Flexibility for Architecting Changeable Systems. In: Conference on Systems Engineering Research (2008)
[22] Steward, D.V.: The design structure system: A method for managing the design of complex systems. IEEE Transactions on Engineering Management EM-28(3), 71–74 (1981), doi:10.1109/TEM.1981.6448589
[23] Suh, E.S., De Weck, O., Chang, D.: Flexible product platforms: framework and case study. Res. Eng. Design 18(2), 67–89 (2007), doi:10.1007/s00163-007-0032-z
[24] Ulrich, K.: The role of product architecture in the manufacturing firm. Research Policy 24(3), 419–440 (1995), doi:10.1016/0048-7333(94)00775-3
[25] Weber, C.: What Is 'Complexity'? In: 15th International Conference on Engineering Design: Engineering Design and the Global Economy, ICED 2005. Engineers Australia, Barton, A.C.T. (2005)
[26] Weinberg, G.M.: An introduction to general systems thinking. Wiley, New York (1975)

User-Based Solutions for Increasing Level of Service in Bike-Sharing Transportation Systems

Juste Raimbault

Abstract. Bike-sharing transportation systems have been well studied from a top-down viewpoint, either for an optimal conception of the system, or for a better statistical understanding of their working mechanisms in the aim of the optimization of the management strategy. Yet bottom-up approaches that could include behavior of users have not been well studied so far. We propose an agent-based model for the short time evolution of a bike-sharing system, with a focus on two strategical parameters that are the role of the quantity of information users have on the all system and the propensity of user to walk after having dropped their bike. We implement the model in a general way so it is applicable to every system as soon as data are available in a certain format. The model of simulation is parametrized and calibrated on processed real time-series of bike movements for the system of Paris. After showing the robustness of the simulations by validating internally and externally the model, we are able to test different user-based strategies for an increase of the level of service. In particular, we show that an increase of user information can have significant impact on the homogeneity of repartition of bikes in docking stations, and, what is important for an future implementation of the strategy, that an action on only 30% of regular users is enough to obtain most of the possible amelioration.

Keywords: bike-sharing transportation system, agent-based modeling, bottom-up complex system management.

Juste Raimbault
Graduate School, Ecole Polytechnique, Palaiseau, France and
LVMT, Ecole Nationale des Ponts et Chaussées,
Champs-sur-Marne, France
e-mail: juste.raimbault@polytechnique.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_3

1 Introduction

Bike-sharing transportation systems have been presented as an ecological and user-friendly transportation mode, which appears to be well complementary to classic public transportation systems ([1]). The quick propagation of many implementations of such systems across the world confirms the interesting potentialities that bike-sharing can offer [2]. O'Brien & *al.* propose in [3] a review on the current state of bike-sharing across the world. Inspired by the relatively good success of such systems in Europe, possible key factors for their quality have been questioned and transposed to different potential countries such as China ([4, 5]) or the United States ([6]).

The understanding of system mechanisms is essential for its optimal exploitation. That can be done through statistical analysis with predictive statistical models ([7–10]) or data-mining techniques ([3, 11]), and can give broader results such as structure of urban mobility patterns. Concerning the implementation, a crucial point in the design of the system is an optimal location of stations. That problem have been extensively studied from an Operational Research point of view ([12, 13] for example). The next step is a good exploitation of the system. By nature, strong asymmetries appear in the distribution of bikes: docking stations in residential areas are emptied during the day contrary to working areas. That causes in most cases a strong decrease in the level of service (no parking places or no available bikes for example). To counter such phenomena, operators have redistribution strategies that have also been well studied and for which optimal plans have been proposed ([14–16]).

However, all these studies always approach the problem from a top-down point of view, in the sense of a centralized and global approach of the issues, whereas bottom-up strategies (i. e. local actions that would allow the emergence of desired patterns) have been to our knowledge not much considered in the literature. User-based methods have been considered in [17, 18] in the case of a car-sharing system, but the problem stays quite far from a behavioral model of the agents using the system, since it explores the possibility of implication of users in the redistribution process, or of shared travels what is not relevant in the case of bikes. Indeed the question of a precise determination of the influence of users behaviors and parameters on the level of service of a bike-sharing systems remains open. We propose an agent-based model of simulation in order to represent and simulate the system from a bottom-up approach, considering bikers and parking as stations as agents and representing their interactions and evolutions in time. That allows to explore user-targeted strategies for an increase of the level of service, as the incitation to use online information media or to be more flexible on the destination point. Note that our work aims to explore effects of user-based policies, but does not pretend to give recommendations to system managers, since our approach stays technical and eludes crucial political and human aspects that

one should take into account in a broader system design or management context.

The rest of the paper is organized as follows. The model and indicator used to quantify its behavior are described in Section 2. Next, Section 3 presents the implementation and results, including internal and external validations of the model by sensitivity analysis and simplified calibration on real data, and also exploration of possible bottom-up strategies for system management. We conclude by a discussion on the applicability of results and on possible developments.

2 Presentation of the Model

Introduction. The granularity of the model is the scale of the individual biker and of the stations where bikes are parked. A more integrated view such as flows would not be useful to our purpose since we want to study the impact of the behavior of individuals on the overall performance of the system. The global working scheme consists in agents embedded in the street infrastructure, interacting with particular elements, what is inspired from the core structure of the Miro model ([19]). Spatial scale is roughly the scale of the district; we don't consider the whole system for calculation power purposes (around 1300 stations on all the system of Paris, whereas an interesting district have around 100 stations), what should not be a problem as soon as in- and outflows allow to reconstruct travels entering and getting out of the area. Tests on larger spatial zones showed that generated travel were quite the same, justifying this choice of scale. Focusing on some particular districts is important since issues with level of service occur only in narrow areas. Time scale of a run is logically one full day because of the cyclic nature of the process ([20]).

Formalisation. The street network of the area is an euclidian network ($V \subset \mathbb{R}^2, E \subset V \times V$) in a closed bounded part of $\mathbb{R}^2$. The time is discretized on a day, so all temporal evolution are defined on $T = [0, 24] \cap \tau\mathbb{N}$ with τ time step (in hours). Docking stations S are particular vertices of the network for which constant capacities $c(s \in S)$ are defined, and that can contain a variable number of bikes $p_b(s) \in \{0, \ldots, c\}^T$. We suppose that temporal fields $O(x, y, t)$ and $D(x, y, t)$ are defined, corresponding respectively to probabilities that a given point at a given time becomes the expected departure (resp. the expected arrival) of a new bike trip, knowing that a trip starting (resp. arriving) at

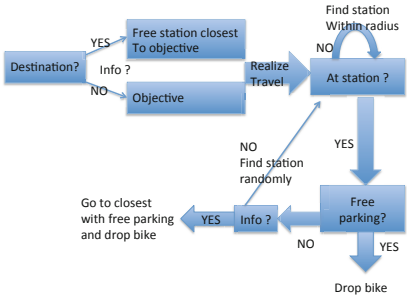


Fig. 1 Flowchart of the decision process of bikers, from the start of their travel to the drop of the bike.

that time exists. Boundaries conditions are represented as a set of random variables $(N_I(i,t))$. For each possible entry point $i \in I$ ($I \subset V$ is a given set of boundaries points) and each time, $N_I(i,t)$ gives the number of bikes trips entering the zone at point i and time t. For departures, a random time-serie $N_D(t)$ represents the number of departures in the zone at time t. Note that these random variables and probabilities fields are sufficient to built the complete process of travel initiation at each time step. Parametrization of the model will consist in proposing a consistent way to construct them from real data.

Docking stations are fixed agents, only their functions p_b will vary through time. The other core agents are the bikers, for which the set $B(t)$ is variable. A biker $b \in B(t)$ is represented by its mean speed $\bar{v}(b)$, a distance $r(b)$ corresponding to its "propensity to walk" and a boolean $i(b)$ expressing the capacity of having access to information on the whole system at any time (through a mobile device and the dedicated application for example). The initial set of bikers $B(0)$ is taken empty, as $t = 0$ corresponds to 3a.m. when there is approximately no travels on standard days.

We define then the workflow of the model for one time step. The following scheme is sequentially executed for each $t \in T$, representing the evolution of the system on a day.

For each time step the evolution of the system follows this process :

- Starting new travels. For a travel within the area, if biker has information, he will adapt his destination to the closest station of its destination with free parking places, if not his destination is not changed.
  - For each entry point, draw number of new traveler, associate to each a destination according to D and characteristics (information drawn uniformly from proportion of information, speed according to fixed mean speed, radius also).
  - Draw new departures within the area according to O, associate either destination within (in proportion to a fixed parameter p_{it}, proportion of internal travels) the area, or a boundary point (travel out of the area). If the departure is empty, biker walks to an other station (with bikes if has information, a random one if not) and will start his travel after a time determined by mean walking speed and distance of the station.
  - Make bikers waiting for start for which it is time begin their journey (correspond to walkers for which a departure station was empty at a given time step before)
- Make bikers advance of the distance corresponding to their speed. Travel path is taken as the shortest path between origin and destination, as effective paths are expected to have small deviation from the shortest one in urban bike travels [8].
- Finish travels or redirect bikers

- if the biker was doing an out travel and is on a boundary point, travel is finished (gets out of the area)
- if has no information, has reached destination and is not on a station, go to a random station within $r(b)$
- if is on a station with free places, drop the bike
- if is on a station with no places, choose as new destination either the closest station with free places if he has information, or a random one within $r(b)$ (excluding already visited ones, implying the memory of agents).

Fig. 1 shows the decision process for starting and arriving bikers. Note that walking radius $r(b)$ and information $i(b)$ have implicitly great influence on the output of the model, since dropping station is totally determined (through a random process) by these two parameters when the destination is given.

Evaluation Criteria. In order to quantify the performance of the system, to compare different realizations for different points in the parameter space or to evaluate the fitness of a realization towards real data, we need to define some functions of evaluation, proxies of what are considered as "qualities" of the system.

Temporal Evaluation Functions. These are criteria evaluated at each time step and for which the output on the all shape of the time-series will be compared.

- Mean load factor $\bar{l}(t) = \frac{1}{|S|} \sum_{s \in S} \frac{p_b(s)}{c(s)}$
- Heterogeneity of bike distribution: we aggregate spatial heterogeneity of load factors on each station through a standard normalized heterogeneity indicator, defined by $h(t) = \frac{2}{\sum_{s \neq s' \in S} \frac{1}{d(s,s')}} \cdot \sum_{s \neq s? \in S} \frac{\left| \frac{p_b(s,t)}{c(s)} - \frac{p_b(s',t)}{c(s')} \right|}{d(s,s')}$

Aggregated Evaluation Functions. These are criteria aggregated on a all day quantifying the level of service integrated on all travels. We note $\mathcal{T}$ the set of travels for a realization of the system and $\mathcal{A}$ the set of travel for which an "adverse event" occured, i. e. for which a potential dropping station was full or a starting station was empty. For any travel $v \in \mathcal{T}$, we denote by $d_{th}(v)$ the theoretical distance (defined by the network distance between origin and initial destination) and $d_r(v)$ the effective realized distance.

- Proportion of adverse events: proportion of users for which the quality of service was doubtful. $A = \frac{|\mathcal{A}|}{|\mathcal{T}|}$
- Total quantity of detours: quantification of the deviation regarding an ideal service $D_{tot} = \frac{1}{|\mathcal{T}|} \cdot \sum_{v \in \mathcal{T}} \frac{d_r(v)}{d_{th}(v)}$

We also define a fitness function used for calibration of the model on real data. If we note $(lf(s,t))_{s \in S, t \in T}$ the real time-series extracted for a standard day by a statistical analysis on real data, we calibrate on the mean-square error on all time-series, defined for a realization of the model by

$$MSE = \frac{1}{|S|\,|T|}\sum_{t\in T}\sum_{s\in S}\left(\frac{p_b(s,t)}{c(s)} - lf(s,t)\right)^2$$

3 Results

3.1 Implementation and Parametrization

Implementation. The model was implemented in NetLogo ([21]) including GIS data through the GIS extension. Preliminary treatment of GIS data was done with QGIS ([22]). Statistical pre-treatment of real temporal data was done in R ([23]), using the NL-R extension ([24]) to import directly the data. For complete reproducibility, source code (including data collection scripts, statistical R code and NetLogo agent-based modeling code) and data (raw and processed) are available on the open git repository of the project at `http://github.com/JusteRaimbault/CityBikes`.

Concerning the choice of the level of representation in the graphical interface, we followed Banos in [25] when he argues that such exploratory models can really be exploited only if a feedback through the interface is possible. It is necessary to find a good compromise for the quantity of information displayed in the graphical interface. In our case, we represent a map of the

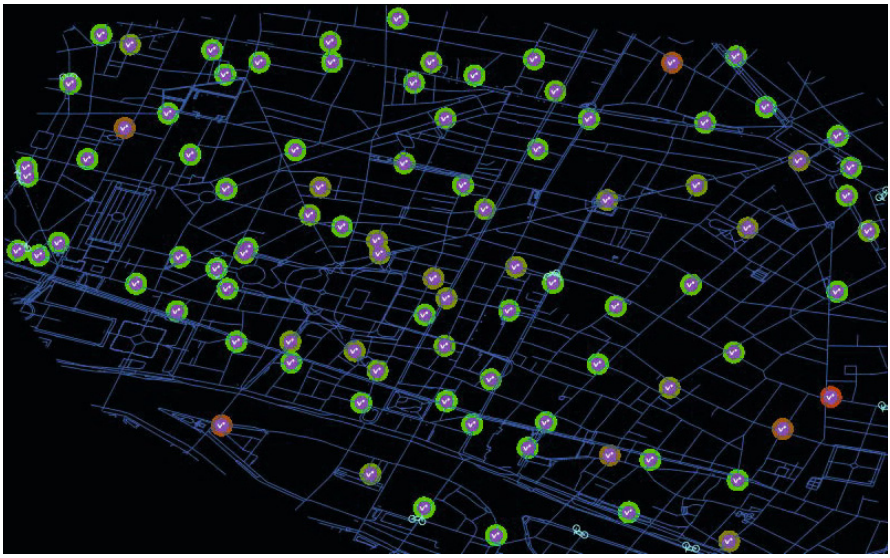


Fig. 2 Example of the graphical output of the model for a particular district (Chatelet). The map shows docking stations, for each the color gradient from green to red gives the current loading factor (green : empty, red : full).

district, on which link width is proportional to current flows, stations display their load-factor by a color code (color gradient from green, $lf(s) = 0$, to red, $lf(s) = 1$). Bikes are also represented in real time, what is interesting thanks to an option that allow to follow some individuals and visualize their decision process through arrows representing original destination, provenance and new destination (should be implemented in further work). This feature could be seen as superficial at this state of the work but it appears as essential regarding possible further developments of the project (see discussion section). Fig. 2 shows an example of the graphical interface of the implementation of the model of simulation.

Data Collection. All used data are open data, in order to have good reproducibility of the work. Road network vector layer was extracted from OpenStreetMap ([26]). Time-series of real stations statuts for Paris were collected automatically[1] all 5 minutes during 6 month and were imported into R for treatment with [27] and the point dataset of stations was created from the geographical coordinates with [28].

Parametrization. The model was designed in order to have real proxies for most of parameters. Mean travel speed is taken as $\bar{v}$ =14km.h^{-1} from [29], where data of trips where studied for the bike system of the city of Lyon, France. To simplify, we take same speed for all bikers : $v(b) = \bar{v}$. A possible extension with tiny gaussian distribution around mean speed showed in experiments to bring nothing more. It has been shown in [3] that profiles of use of bike systems stays approximatively the same for european cities (but can be significantly different for cities as Rio or Taipei), what justify the use of these inferred data in our case. We also use the determined mean length of travel from [16] (here that parameter should be more sensible to the topology so we prefer extract it from this second paper although it seems to have subsequent methodological bias compared to the first rigorous work on the system of Lyon), which is 2.3km, in order to determine the diameter of the area on which our approach stays consistent. Indeed the model is built in order to have emphasis on travels coming from the outside and on travels going out, internal travels have to stay a small proportion of all travels. In our case, a district of diameter 2km gives a proportion of internal travels $p_{it} \approx 20\%$. We will take districts of this size with this fixed proportion in the following.

The crucial part of the parametrization is the construction of O, D fields and random variables N_I, N_D from real data. Daily data were reduced through sampling of time-series of load-factors of all stations and dimension of the representation of a day was significantly reduced through a k-means

[1] From the dedicated website `api.jcdecaux.com`

clustering procedures (classically used in time-series clustering as it is described in [30]). These reduced points were then clustered again in order to isolate typical weekdays from week-ends, where the use profiles are typically different and from special days such as ones with very bad climate or public transportation strikes. That allowed to create the profile of a "standard day" that was used to infer O, D fields through a spatial Gaussian multi-kernel estimation (see [31]). The characteristic size of kernels $1/\sigma$ is an essential parameter for which we have no direct proxy, and that will have to be fixed through a calibration procedure. The laws for N_I, N_D were taken as binomial: for an actual arrival, we consider each possible travel and increase the number of drawing of each binomial law of entries by 1 at the time corresponding to mean travel time (depending on the travel distance) before arrival time. Probabilities of binomial laws are $1/Card(I)$ since we assume independence of travels. For departure, we just increase by one drawings of the binomial law at current time for an actual departure.

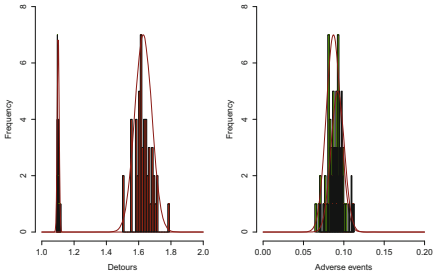


Fig. 3 Statistical analysis of outputs. For some aggregated outputs (here the overall quantity of detours and the proportion of adverse events), we plotted histograms of the statistical distribution of the functions on many realizations of the model for a point in the parameter space. Two points of the parameter space, corresponding to $(r = 300, p_{info} = 50, \sigma = 80)$ (green histogram) and $(r = 700, p_{info} = 50, \sigma = 80)$ (red) are plotted here as examples. Gaussian fits are also drawn. The relative good fit shows the internal consistence of the model and we are able to quantify the typical number of repetitions needed when applying the model : supposing normal distributions for the indicator and its mean, a 95% confidence interval of size $\sigma/2$ is obtained with $n = (2 \cdot 2\sigma \cdot 1.96/\sigma)^2 \approx 60$

What we call parameter space in the following consists in the 3 dimensional space of parameters that have not been fixed by this parametrization, i. e. the walking radius r (taken as constant on all bikers, as for the speed), the information proportion p_{info} what is the probability for a new biker to have information and the "size" of the Gaussian kernels σ (note that the spread of distributions is decreasing with σ).

3.2 Robustness Assessment, Exploration and Calibration

Internal Consistence of the Model. Before using simulations of the model to explore possible strategies, it is necessary to assess that the results produced are internally consistent, i. e. that the randomness introduced in the parametrization and in the internal rules do not lead to divergences in results. Simulations were launched on a large number of repetitions for

different points in the parameter space and statistical distribution of aggregated outputs were plotted. Fig. 3 shows example of these results. The relative good gaussian fits and the small deviation of distributions confirm the internal consistence of the model. We obtain the typical number of repetitions needed to have a 95% confidence interval of length half of the standard deviation, what is around 60, and we take that number in all following experiments and applications. These experiments allowed a grid exploration of the parameter space, confirming expected behavior of indicators. In particular, the shape of MSE suggested to use the simplified calibration procedure presented in the following.

Robustness Regarding the Study Area. The sensitivity of the model regarding geometry of the area was also tested. Experiments described afterwards were run on comparable districts (Châtelet, Saint-Lazare and Montparnasse), leading to the same conclusions, what confirms the external robustness of the model.

Reduced Calibration Procedure. Using experiments launched during the grid exploration of the parameter space, we are able to assess or the regularity of some aggregated criteria, especially of the mean-square error on loads factors of stations. We calibrate on kernel size and quantity of information. For different values of the walking radius, the obtained area for the location of the minimum of the mean-square error stays quite the same for reasonable values of the radius (300-600m). Fig. 4 shows an example of the surface used for the simplified calibration. We extract from that the values of around 50 for kernel size and 30 for information proportion. The most important is kernel size since we cannot have real proxy for that parameter. We use these values for the explorations of strategies in the following.

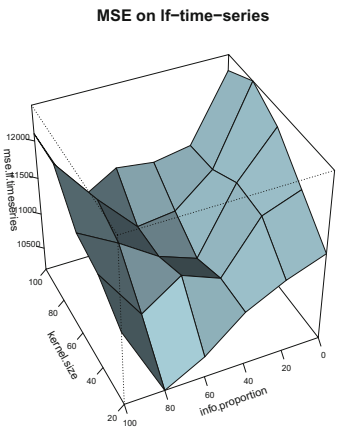
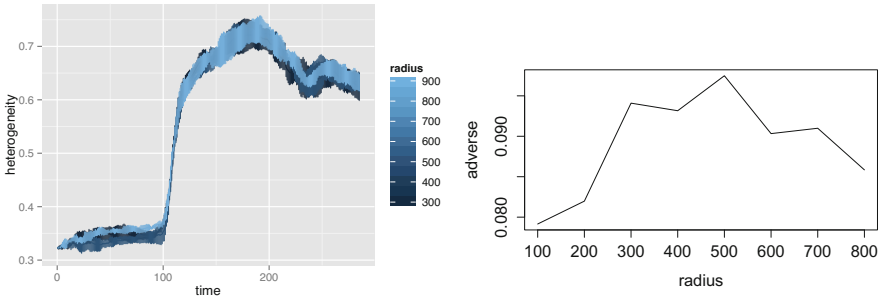


Fig. 4 Simplified calibration procedure. We plot the surface of the mean-square error on time-series of load-factors as a function of the two parameters on which we want to calibrate. For visibility purpose, only one surface was represented out of the different obtained for different values of walking radius. The absolute minimum obtained for very large kernel has no sense since such value give quasi-uniform probabilities because of total recovering of Gaussian kernels. We take as best realization the second minimum, which is located around a kernel size of 50 and a quantity of information of 30%, which seem to be reasonable values afterwards.

3.3 Investigation of User-Based Strategies

Influence of Walking Radius. Taking for kernel-size and quantity of information the values given by the calibration, we can test the influence of walking radius on the performance of the system. Note that we make a strong assumption, that is that the calibration stay valid for different values of the radius. As we stand previously, this stays true as soon as we stay in a reasonable range of values (we obtained 300m to 600m) for the radius. The influence of variations of walking radius on indicators were tested. Most interesting results are shown in figure 5. Concerning the indicators evaluated on time-series (h and $\bar{l}(t)$), it is hard to have a significant conclusion since the small difference that one can observe between curves lies inside errors bars of all curves. For A, we see a decreasing of the indicator after a certain value (300m), what is significant if we consider that radius under that value are not realistic, since a random place in the city should be at least in mean over 300m from a bike station. However, the results concerning the radius are not so concluding, what could be due to the presence of periodic negative feedbacks: when the mean distance between two stations is reached, repartitions concerns neighbor stations as expected, but the relation is not necessarily positive, depending on the current status of the other station. A deeper understanding and exploration of the behavior of the model regarding radius should be the object of further work.

Influence of Information. For the quantity of information, we are on the contrary able to draw significant conclusions. Again, behavior of indicators were studied according to variations of p_{info}. Most significant are shown on



(a) Time series of heterogeneity indicator $h(t)$ for different values of walking radius. Small differences between means could mislead to a positive effect of radius on heterogeneity, but the error bars of all curves recover themselves, what makes any conclusion non-significant.

(b) Influence of walking radius on the quantity of adverse events A. After 400m, we observe a relative decrease of the proportion. However, values under 300-400m should be ignored since these are smaller than the mean distance of a random point to a station.

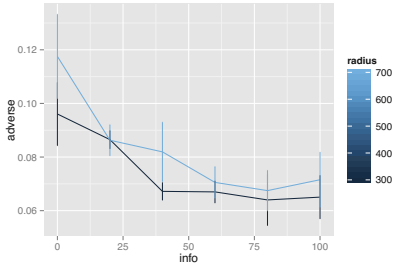
Fig. 5 Results on the influence of walking radius

figure 6. Results from time-series are also not concluding, but concerning aggregated indicators, we have a constant and regular decrease for each and for different values of the radius. We are able to quantify a critical value of the information for which most of the progress concerning indicator A (adverse events) is done, that is around 35%. We observe for this value an amelioration of 4% in the quantity of adverse events, that is interesting when compared to the total number of bikers. Regarding the management strategy for an increase in the level of service, that implies an increase of the penetration rate of online information tools (mobile application e. g.) if that rate is below 50%. If it is over that value, we have shown that efforts for an increase of penetration rate would be pointless.

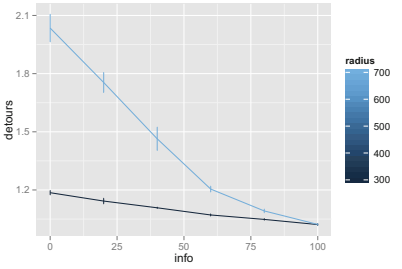
4 Discussion

4.1 Applicability of the Results

We have shown that increases of both walking radius and information quantity could have positive consequences on the level of service of the system, by reducing the overall number of adverse events and the quantity of detours especially in the case of the information. However, we can question the possible applicability of the results. Concerning walking radius, first a deeper investigation would be needed for confirmation of the weak tendency we observed, and secondly it appears that in reality, it should be infeasible to play on that parameter. The only way to approach that would be to inform users of the



(a) Influence of proportion of information on adverse events A for two different values of walking radius. We can conclude significantly that the information has a positive influence. Quantitatively, we extract the threshold of 35% that corresponds to the majority of decrease, that means that more is not necessarily needed.



(b) Influence of information on quantity of detours D_{tot}. Curves for $r = 300m$ and $r = 700m$ are shown (scale color). Here also, the influence is positive. The effect is more significant for high values of walking radius. The inflection is around 50% of users informed, what is more than for adverse events.

Fig. 6 Results on the influence of proportion of information.

potential increase in the level of service if they are ready to make a little effort, but that is quite optimistic to think that they will apply systematically the changes, either because they are egoistic, because they won't think about it, or because they will have no time.

Concerning the information proportion, we cannot also force users to have information device (although a majority of population owns such a device, they won't necessarily install the needed software, especially if that one is not user-friendly). We should proceed indirectly, for example by increasing the ergonomics of the application. An other possibility would to improve information displayed at docking stations that is currently difficult to use.

4.2 Possible Developments

Other Possible Management Strategies. Concerning user parameters, other choices could have been made, as including waiting time at a fixed place, either for a parking or a bike. The parameters chosen are both possible to influence and quite adapted to the behavioral heuristic used in the model, and therefore were considered. Including other parameters, or changing the behavioral model such as using discrete choice models may be possible developments of our work. Furthermore, only the role of user was so far explored. The object of further investigation could be the role of the "behavior" of docking stations. For example, one could fix rules to them, as close all parkings over a certain threshold of load-factor, or allow only departures or parkings in given configurations, etc. Such intelligent agents would surely bring new ways to influence the overall system, but will also increase the level of complexity (in the sense of model complexity, see [32]), and therefore that extension should be considered very carefully (that is the reason why we did not integrate it in this first work).

Towards an Online Bottom-up Pilotage of the Bike-Sharing System. Making the stations intelligent can imply making them communicate and behave as a self-adapting system. If they give information to the user, the heterogeneity of the nature and quantity of information provided could have strong impact on the overall system. That raises of course ethical issues since we are lead to ask if it is fair to give different quantities of information to different users. However, the perspective of a bottom-up piloted system could be of great interest from a theoretical and practical point of view. One could think of online adaptive algorithms for ruling the local behavior of the self-adapting system, such as ant algorithms ([33]), in which bikers would depose virtual pheromones when they visit a docking station (corresponding to their information on travel that is easy to obtain), that would allow the system to take some local decisions of redirecting bikers or closing stations for a short time in order to obtain an overall better level of service. Such methods have already been studied to improve level of service in other public transportation systems like buses [34].

5 Conclusion

This work is a first step of a new bottom-up approach of bike-sharing systems. We have implemented, parametrized and calibrated a basic behavioral model and obtained interesting results for user-based strategies for an increase of the level of service. Further work will consist in a deeper validation of the model, its application on other data. We suggest also to explore developments such as extension to other types of agents (docking stations), or the study of possible bottom-up online algorithm for an optimal pilotage of the system.

References

1. Midgley, P.: The role of smart bike-sharing systems in urban mobility. Journeys 2, 23–31 (2009)
2. DeMaio, P.: Bike-sharing: History, impacts, models of provision, and future. Journal of Public Transportation 12(4), 41–56 (2009)
3. O'Brien, O., Cheshire, J., Batty, M.: Mining bicycle sharing data for generating insights into sustainable transport systems. Journal of Transport Geography (2013)
4. Liu, Z., Jia, X., Cheng, W.: Solving the last mile problem: Ensure the success of public bicycle system in beijing. Procedia-Social and Behavioral Sciences 43, 73–78 (2012)
5. Geng, X., Tian, K., Zhang, Y., Li, Q.: Bike rental station planning and design in paris. Urban Transport of China 4, 008 (2009)
6. Gifford, J.: Will smart bikes succeed as public transportation in the united states? Center for Urban Transportation Research 7(2), 1 (2004)
7. Borgnat, P., Abry, P., Flandrin, P.: Modélisation statistique cyclique des locations de vélo'v à lyon. In: XXIIe Colloque GRETSI (traitement du signal et des images), Dijon (FRA), Septembre 8-11. GRETSI, Groupe d'Etudes du Traitement du Signal et des Images (2009)
8. Borgnat, P., Fleury, E., Robardet, C., Scherrer, A., et al.: Spatial analysis of dynamic movements of vélo'v, lyon's shared bicycle program. In: European Conference on Complex Systems 2009 (2009)
9. Borgnat, P., Abry, P., Flandrin, P., Rouquier, J.-B., et al.: Studying lyon's vélo'v: a statistical cyclic model. In: European Conference on Complex Systems 2009 (2009)
10. Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.-B., Fleury, E.: Shared bicycles in a city: A signal processing and data analysis perspective. Advances in Complex Systems 14(03), 415–438 (2011)
11. Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J., Banchs, R.: Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. Pervasive and Mobile Computing 6(4), 455–466 (2010)
12. Lin, J.-R., Yang, T.-H., Chang, Y.-C.: A hub location inventory model for bicycle sharing system design: Formulation and solution. Computers & Industrial Engineering (2011)
13. Lin, J.-R., Yang, T.-H.: Strategic design of public bicycle sharing systems with service level constraints. Transportation Research Part E: Logistics and Transportation Review 47(2), 284–294 (2011)

14. Kek, A.G.H., Cheu, R.L., Chor, M.L.: Relocation simulation model for multiple-station shared-use vehicle systems. Transportation Research Record: Journal of the Transportation Research Board 1986(1), 81–88 (2006)
15. Nair, R., Miller-Hooks, E.: Fleet management for vehicle sharing operations. Transportation Science 45(4), 524–540 (2011)
16. Nair, R., Miller-Hooks, E., Hampshire, R.C., Bušić, A.: Large-scale vehicle sharing systems: Analysis of vélib'. International Journal of Sustainable Transportation 7(1), 85–106 (2013)
17. Barth, M., Todd, M.: Simulation model performance analysis of a multiple station shared vehicle system. Transportation Research Part C: Emerging Technologies 7(4), 237–259 (1999)
18. Barth, M., Todd, M., Xue, L.: User-based vehicle relocation techniques for multiple-station shared-use vehicle systems. TRB Paper No. 04-4161 (2004)
19. Banos, A., Boffet-Mas, A., Chardonnel, S., Lang, C., Marilleau, N., Thévenin, T., et al.: Simuler la mobilité urbaine quotidienne: le projet miro. Mobilités urbaines et risques des transports (2011)
20. Vogel, P., Greiser, T., Mattfeld, D.C.: Understanding bike-sharing systems using data mining: Exploring activity patterns. Procedia-Social and Behavioral Sciences 20, 514–523 (2011)
21. Wilensky, U.: Netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL (1999)
22. QGIS Development Team. QGIS Geographic Information System. Open Source Geospatial Foundation (2009)
23. R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2013)
24. Thiele, J.C., Kurth, W., Grimm, V.: Agent-based modelling: Tools for linking netlogo and r. Journal of Artificial Societies and Social Simulation 15(3), 8 (2012)
25. Banos, A.: Pour des pratiques de modélisation et de simulation libérées en Géographie et SHS. PhD thesis, UMR CNRS Géographie-Cités, ISCPIF (Décembre 2013)
26. Bennett, J.: OpenStreetMap. Packt Publishing (2010)
27. Couture-Beil, A.: rjson: Json for r. R package version 0.2, 13 (2013)
28. Timothy, H.: Keitt, Roger Bivand, Edzer Pebesma, and Barry Rowlingson. rgdal: bindings for the geospatial data abstraction library (2011), R package version 0.7-1, `http://CRAN.R-project.org/package=rgdal`
29. Jensen, P., Rouquier, J.-B., Ovtracht, N., Robardet, C.: Characterizing the speed and paths of shared bicycle use in lyon. Transportation Research Part D: Transport and Environment 15(8), 522–524 (2010)
30. Liao, T.W.: Clustering of time series data—a survey. Pattern Recognition 38(11), 1857–1874 (2005)
31. Alexandre, B.: Tsybakov. Introduction to nonparametric estimation (introduction à l'estimation non-paramétrique) (2004)
32. Varenne, F., Silberstein, M., et al.: Modéliser & simuler. Epistémologies et pratiques de la modélisation et de la simulation, tome 1 (2013)
33. Monmarché, N.: Algorithmes de fourmis artificielles: applicationsa la classification eta l'optimisation. PhD thesis, École Polytechnique (2004)
34. Gershenson, C.: Self-organization leads to supraoptimal performance in public transportation systems. PLoS ONE 6(6), e21469 (2011)

A New Framework for the Simulation of Offshore Oil Facilities at the System Level

Marc Bonnissel, Joris Costes, Jean-Michel Ghidaglia, Philippe Muguerra, Keld Lund Nielsen, Benjamin Poirson, Xavier Riou, Jean-Philippe Saut, and Nicolas Vayatis

Abstract. Offshore oil facilities are complex industrial systems: They are composed of numerous parts and involve both elaborate physics and stochastic aspects like failure risk or price variation. Several software tools are available to simulate individual components of offshore facilities, for instance to compute the flow dynamics in a particular device. There is however no tool to simulate the facility at the system level, i.e. to simulate the general behavior of the facility. The paper presents a framework for such a system-level simulator, which includes one layer for physics and one for risk simulation. The physical part uses the equation-based language Modelica[1]. Modelica components are defined to model typical devices of an installation. The risk simulation uses Markov chains and statistical indicators to assess performance and resilience of the system. It runs with an external language (C or Scilab) and data from the Modelica simulation.

Keywords: system-level simulation, fluid flow, two-phase flow, risk estimation, Monte Carlo simulation.

Marc Bonnissel · Philippe Muguerra · Benjamin Poirson · Xavier Riou
Saipem s.a., 7 avenue de San Fernando, 78180 Montigny-le-Bretonneux, France
e-mail: {Marc.BONNISSEL,Philippe.MUGUERRA,Benjamin.POIRSON,
Xavier.RIOU}@saipem.com

Joris Costes · Jean-Philippe Saut
Eurobios SCB, 86 avenue Lénine, 94250 Gentilly, France
e-mail: {joris.costes,jpsaut}@eurobios.com

Joris Costes · Jean-Michel Ghidaglia · Nicolas Vayatis
CMLA, ENS Cachan, UMR 8536 CNRS, 61 avenue du président Wilson, 94235 Cachan cedex, France
e-mail: {jmg,vayatis}@cmla.ens-cachan.fr

Keld Lund Nielsen
Eni S.p.A., exploration & production division, San Donato Milanese (Mi), Italy
e-mail: Keld.Lund.Nielsen@eni.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_4

1 Introduction

Designing an offshore oil facility is a complicated task, which requires a strong expertise in various fields like fluid dynamics, control systems or system reliability. Considering the related investments and safety concerns, the consequences of a bad design can be critical. From this view, computer simulation is a very valuable tool as it can be used to test an installation at almost no cost –once it is available, i.e. without considering development and validation investments– and without any risk. Software tools are available to simulate individual components of an offshore facility, for instance to compute the flow dynamics in a specific device. Some recent simulators can also simulate the fluid flow in the whole facility. However there is still no tool to simulate the facility at the system level. System-level simulation aims at considering all that can influence the system behavior during its life-time. This means considering not only physics but also phenomena like failure events, maintenance, economic factors or weather conditions. Simulating at the system level is useful to study the overall behavior of the facility. Besides, different accuracy levels make possible the simulation of the whole installation over its whole life-cycle as well as finer simulations, over a shorter time period or limited areas. Low accuracy levels require less computation time but are also useful to bring simulation into play earlier in the design process, when not-yet-complete specifications do not allow the use of fine models. System-level simulation of offshore facilities could certainly be very useful to engineers or decision makers for specification, design and study. Having a common tool for cross-discipline experts and managers could greatly enhance the cooperative efforts and communication.

The development of a system-level simulation for offshore oil facilities is the motivation of the FASTPROOF project. The project is a collaboration between industrial (eurobios, eni, eni saipem) and an academic partner, CMLA (ENS Cachan and CNRS). Figure 1 shows a simplified example of offshore facility: Several sub-sea wellheads are connected to an oil reservoir; The oil and gas mixture flows from the wells to pipelines; The mixture is then collected via a manifold and flows up through a vertical pipe (*riser*) into the FPSO (*Floating Production, Storage and Offloading*) unit, where oil is separated from gas, in order to be stored in a tank. Other kinds of component are present in offshore installation, e.g. pumps, lifters or electrical supply. The choice of the components and how they are connected together defines a design. The simulation tool should be able to let the user select and connect the components freely, in order to test any design. This requires a modular modeling, which is possible if the modeling language offers specific features like object-oriented programming and acausal modeling. Modelica is a simulation language with such capabilities. Section 2 presents some principles of acausal modeling and its implementation in Modelica. Modelica was designed to model deterministic systems, from differential equations, with a focus on physical systems. Section 3 presents the different physical phenomena of interest in an offshore

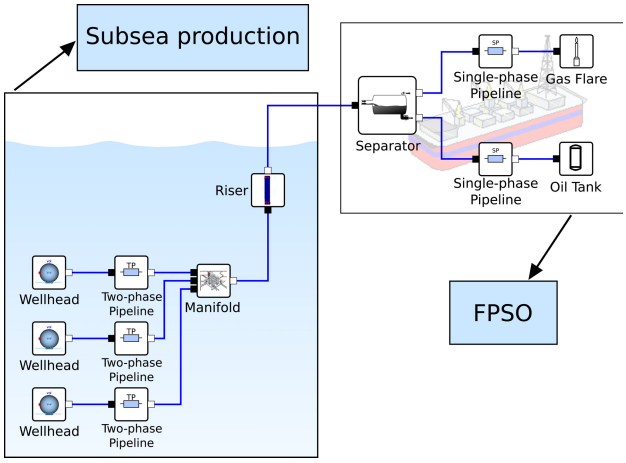


Fig. 1 Overview of an offshore oil production installation. Each block icon corresponds to a component or subsystem.

facility. Over a long time-period however, a purely deterministic simulation is not relevant since many random events can occur (failures of components, degraded weather condition, change of an economic parameter). Stochastic models become then necessary. Section 4 deals with these models, which can be related to risk of various natures. Finally, Section 5 presents the software framework we use to combine physical and risk simulations.

2 Acausal and Hybrid Modeling

2.1 Acausal Modeling

A straightforward way to model physical systems is the so-called causal scheme. A causal model first requires the definition of the system inputs and outputs. Then a relation is used to compute the outputs from the inputs and possible state variables. The relation is typically the integration of a differential equation. This is computationally very efficient because the data flow is explicit. However since the data flow is specified by the user it usually makes modeling difficult. Besides, as the causality of each component is fixed, it is not possible to re-use components or to change the system design without recomputing the whole data flow. In an offshore facility, the flow in a pipe can be the consequence of an imposed difference of pressure at its ends (e.g. pressure in the reservoir and pressure in the tank) or can be imposed by a pump. In the latter case, the friction on the pipe wall will create of difference of pressure between the two pipe ends. The causality relation between flow and pressure difference is then dependent on what is

connected to the pipe. Acausal modeling is the way to represent this principle. Acausal modeling is much closer to the description that could be made by an engineer than causal modeling. Models are symbolic equation sets, not assignations like in causal modeling. An atomic model (or component) is a set of under-determined differential equations. Connecting two components means adding the missing equations (equality of intensive quantities at the connection point, flow quantities summing to zero at the connection point). Other missing equations are set with sources (e.g. source of pressure, thermal energy reservoir). Modelica[1] is a free language, which allows both causal and acausal modeling.

2.2 Modelica

Modelica is a non-proprietary simulation language, actively developed and enriched since 1997. Models are described as differential algebraic equations:

$$\begin{cases} \dot{x}(t) = f(x(t), y(t)), \\ 0 \quad = g(x(t), y(t)), \end{cases} \tag{1}$$

where $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^m$. Only time derivatives are allowed. Space derivatives can be approximated by concatenation of components, e.g. copies of a same pipe model are connected together in a sequence. Objects called connectors (or ports) define the set of variables that obey to specific equations at the physical connections between the components. Connecting components using the language instruction **connect** will automatically add the equations at the connection point. For instance, in fluid applications the most basic connector would be:

```
connector FluidPort
 AbsolutePressure p "Thermodynamic pressure
                     in the connection point";
 flow MassFlowRate m_flow "Mass flow rate from the connection
                           point into the component";
end FluidPort;
```

Connecting port1 of component A to port2 of component B will add the equations:

$$\text{A.port1.p} = \text{B.port2.p}, \tag{2}$$

$$\text{A.port1.m\_flow} + \text{B.port2.m\_flow} = 0. \tag{3}$$

Equations 2 and 3 correspond to the equality of pressure at the connection point, which is an infinitely small volume, and conservation of mass at the connection point. Software tools able to simulate Modelica models are called Modelica tools. They generally propose a graphical user interface so that components are represented by icons and can be dragged and dropped to create a

simulation (Fig. 1). Besides its ability to simulate continuous systems, Modelica can also handle discrete events. Events are conditional changes. They can be used to switch to different equations or to discontinuously reinitialize some quantities, if some particular conditions are met. In oil applications, this could be the formation of chemical substances, like hydrates or wax, in a given pressure-temperature domain.

Modelica offers also many convenient features like an object-oriented design, with an inheritance mechanism, an annotation system for code documentation or clear definitions of variable units.

3 Physics Modeling

Modeling offshore oil production is a multi-physics problem; Fluid dynamics, thermodynamics and chemistry are the fundamental topics but structure mechanics (fatigue) could also be considered. Engineering is also necessary to study controlled systems, power devices or power supply. The simulation is written in Modelica and relies on the concept of components models (under-determined equation sets) and connectors (variables shared between connected components plus equations at the physical connection points).

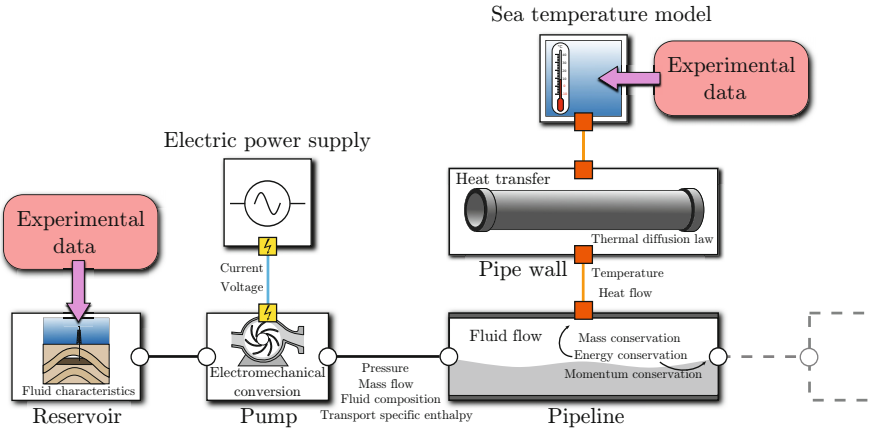


Fig. 2 Some blocks in the physics simulation. Three types of connectors are used: For fluid flow (white circles), for heat transfer (red squares) and for electrical connections (yellow squares with lightning symbol). Each type of connector is associated to physical quantities. Equations inside a component can comprise variables from different connectors. For instance, energy balance equation in the pipe contains a term of heat transfer due to the loss through the thermal connector (thermal diffusion through the pipe wall) and a term of energy transfer due to the fluid flow through the fluid connectors (enthalpy given/received to/from outflowing/inflowing fluid). Some knowledge about the system is purely experimental. In that case, a block that reads data from an experimental database will be used.

Figure 2 shows some typical blocks, equivalent to Modelica models. Some blocks contain equations from one physical domain (electric power supply, heat transfer through pipe wall) and some are at the interface between domains (electro-mechanical devices, thermodynamic models). Modularity of the simulator is clearly visible in Fig. 2: It is straightforward to replace a component with another (for instance different pump models could be tested) or to neglect a physical phenomenon by just disconnecting particular blocks. For instance, disconnecting the pipe-wall block from the pipe-flow block is like assuming a perfectly-insulated pipe.

In the next subsections, we present the categories of components. More detail can be found in [5].

3.1 Fluid Flow

Fluids inside hydrocarbon reservoirs are complex mixtures of numerous substances. Inside the reservoir, gas might be dissolved in oil but, as pressure decreases when the mixture flows towards surface, a gas phase will appear. All flows have consequently to be modeled as two-phase flows, excepted after the separation process, generally performed in surface, aboard the FPSO. Models with different complexities can be built, depending on the assumptions. Some equations are common to all models: Conservation of mass, conservation of momentum, conservation of total energy. If the fluid is compressible, an equation of state is also required. In all cases, accurate prediction is not possible for several reasons:

1. Flow regime in the oil production process are always highly turbulent (Reynolds number $> 10\,000$),
2. Some parameters are not known precisely (e.g. friction coefficient inside the pipes),
3. The system spatial dimensions are huge; lines can be as long as dozens of kilometers.

Therefore it is not realistic to perform detailed computational fluid dynamics with fine space sampling. Instead, the flow models will be 0D and spatial discretization will be approximated. Without writing new equations with space quantities, a 1D model can be approximated if multiple 0D models are connected together. Each model is 0D but the spatial quantity (e.g. the length) influences its behavior. For instance, in the case of a pipe model, the length is taken into account in the computation of the pipe-wall heat-transfer coefficient, in the mass of liquid contained in the pipe and in the pipe-wall friction coefficient. The discretization of a pipe with length L is then obtained by concatenation of n 0D pipe models, each with length $\frac{L}{n}$. n is a parameter of the requested accuracy. The discretization is interesting to visualize the evolution of a quantity along the line, e.g. fluid temperature, which decreases along unheated lines.

3.2 Thermal Transfer

Fluid inside the reservoir is generally at a warm temperature (e.g. 60°C) and is cooled down once flowing in the pipelines because of the low temperature at the sea bottom (e.g. 5°C). Under some P-T conditions chemical substances might start forming (Sect. 3.3), which causes the line to block. Temperature inside the pipes must be computed to detect such formations. In the considered application, thermal transfer is due to thermal diffusion and convection (combination of advective transport and diffusive transport). Inside pipes, heat transfer is mainly due to convection, through its advection term, excepted when a line is blocked (e.g. for maintenance), i.e. when flow is null. Heat transfer with the outside is due to diffusion. Figure 3 shows the different heat transfer modalities. Diffusion is modeled with thermal diffusions laws. The advection term of convection is modeled as a transport enthalpy (specific enthalpy of the fluid instantaneously entering/leaving a pipe) while the diffusion term is neglected.

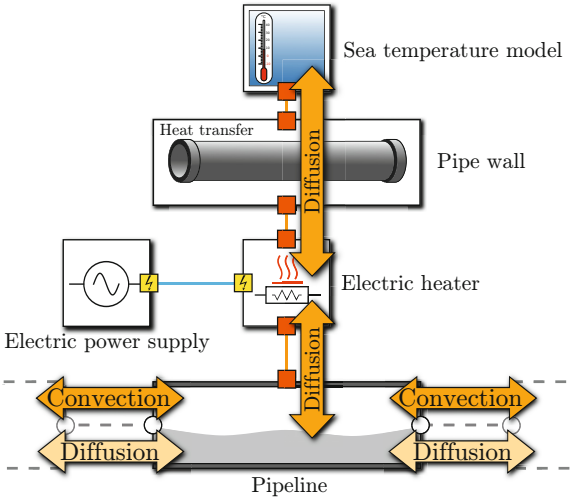


Fig. 3 Possible thermal transfer between some sub-sea components. To maintain temperature inside pipes above hydrate or wax formation temperature, a heating system has been inserted into the pipe. If the fluid is always flowing (i.e. in standard operation), thermal diffusion inside the pipeline can be neglected because convection (through its advection component) is then much more important.

3.3 Hydrate/Wax Formation

Chemical reactions can occur at particular pressure and temperature conditions and lead to the formation of substances that might block lines. Hydrates form when water and natural gas are in contact at high pressure and low temperature. Wax deposition on pipe walls begins if the temperature drops to

the wax appearance point. Fluid velocity and composition condition the formation of hydrates and wax too. The conditions of formation of hydrates and wax are determined from experimental measurements. In the simulator, P-T conditions are monitored and compared to a table to detect when hydrates or wax are susceptible to form.

3.4 Control Systems

Many components are controlled devices. Quantities are regulated by control systems. For instance, the speed of a pump or the gas pressure and liquid level in an oil-gas separator. Figure 4 shows the schematic of a separator. The model contains equations to link the valve apertures to the inlet and outlets flows, to the pressure inside the tank and to the liquid level. The valve apertures are controlled in order to regulate both liquid level and gas pressure. Controllers must then be modeled as well as sensors. Both are causal system, i.e. with specified inputs and outputs. Modelica supports the definition of causal blocks so the whole system can be written as a Modelica component.

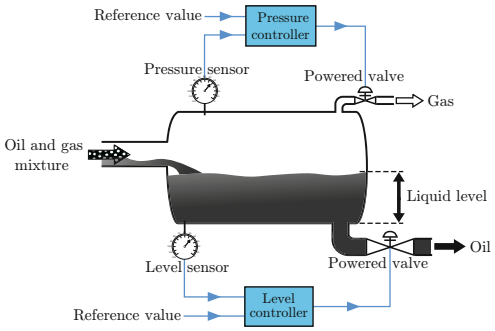


Fig. 4 Example of a controlled system: Oil and gas are separated by gravity; Controllers act on the valve apertures to maintain to reference values both gas pressure and liquid level inside the vessel. Reference values are compared to measured outputs given by sensors, which can also be modeled to take into account their possible imperfections (e.g. non-linearities or temporal drift).

4 Risk Modeling

During the life-time of an exploitation, many factors contribute to the revenue received from the produced oil. We employ the term *risk* to regroup all of the factors that behave with some randomness or cannot be modeled with equations. These factors are modeled as stochastic processes as opposed to the deterministic models of Sect. 3. The risk factors are classified as *endogenous* factors and *exogenous* factors. Exogenous factors occur out of the system-of-interest, whereas endogenous factors occur within the system-of-interest.

4.1 Endogenous Factors

Endogenous factors concern mostly equipment failure events. Building models of equipment failures is the topic of reliability theory. Measured times between failures are collected to build statistical models of failure events. In offshore oil industry, the OREDA project[2] is a major database for failure statistics. Items are organized by type (pump, valve, turbine, etc.) and for each category, failures are classed by severity or modes. Statistics of the failure rate are given for each failure class of each item. If T is the time when a component fails and if $P(T \leq t)$ is the probability that a component fails before a time t, then the failure rate is defined as:

$$\lambda(t) = \frac{f(t)}{R(t)}, \tag{4}$$

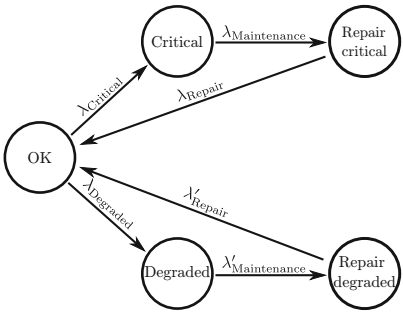


Fig. 5 State machine of the failure and repair states of a single component. The λ terms are the firing rate of each transition. Standard representations like continuous-time Markov chains or Petri nets can be derived from the state machine.

where f is the failure density function[1] and R is the reliability function[2], i.e. the probability that the component does not fail before time t. The failure rate is used to build a probability distribution of the failure time. For instance, in reliability engineering, a common assumption is that λ is constant so the failure time follows an exponential distribution.

The failure model of a component is a state machine. The states are the different failure modes and the transitions from one state to the other are drawn randomly from the failure time distributions. Different representations can be chosen for the failure state machine like Markov chains or Petri nets. To complete the model, the repair process must be clarified. The repair process is related to the maintenance strategy. Some failures, especially critical

[1] The failure density function f is the probability density function of $P(T \leq t)$, i.e. $P(T \leq t) = \int_0^t f(x)dx$.

[2] $R(t) = 1 - P(T \leq t) = P(T > t)$.

failures, might be detected instantaneously by alarm or monitoring systems. The other failures will only be detected at the next maintenance operation. Maintenance operations can be fortuitous or periodically scheduled. Finally, the repair time also needs a model. Databases like OREDA provide some statistics about the repair times for each failure mode. Figure 5 summarizes the different possible states when only two failure modes are considered.

In case some knowledge about dependencies between failures of different components and their joint probability distributions is available, models based on conditional probabilities can be used, e.g. Bayesian Belief Networks[6].

4.2 Exogenous Factors

Exogenous factors do not depend on the installation design or characteristics. They are potentially countless; we focus here on two of them: Price variation and weather conditions.

Price Variation. The net income or gain is the difference between the recovered-oil sale price and the operation and capital costs. Extracted-oil value depends on current crude oil price. Capital cost depends, among other things, on the pipelines cost, which in turn depends on steel price. Crude oil price and steel price both greatly varies with global economy. Price variation cannot be predicted safely. Instead of building predictive models of price variation, it is preferable to include price variation in the risk simulation as scenarios. A scenario is just a predefined record of the price variables as a function of future time. The easiest way to build a scenario is to use historical data, e.g. daily record of the barrel price over a past period of ten years. Scenarios must be chosen in order to represent various and relevant situations, e.g. extreme financial crisis, short-time crisis, stable conditions, etc. The user can then run different simulations, with different designs and price scenarios, to see how each design behaves in the defined scenarios.

Weather Conditions. Sea state (wind, wave and currents) can have a strong impact on installation fatigue and operation tasks. Quantifying this impact is very difficult, if not impossible with the current state-of-the-art. It is more restrictive but preferable to deal with extreme and normal sea-states only. During extreme sea-states, no operation is possible, while during normal sea-states, weather has no impact on operation. An extreme sea-state is defined by thresholds for sea-state-related variables, e.g. wave height. In situ measurements provide sea parameters statistics at the offshore field location. The sea-state is modeled as a Markov chain with two states: Favorable condition and unfavorable condition. The sea state is updated at every time-step (e.g. every three hours) according to the transition probabilities estimated from the measurements. The transition probabilities are not fixed for the whole year but for monthly periods so that seasons are correctly taken into account in the model.

Simulating sea-state can be interesting for instance to find the best date to start the installation of an equipment. If the installation process starts at time T_0 and requires N working days, the best date $\tilde{T_0}$ is the value of T_0 that minimizes ΔT, where ΔT is the time such that there is N working days between T_0 and $T_0 + \Delta T$. The best date $\tilde{T_0}$ is defined for an outcome of the Markov chain simulation. To have a statistic optimal value, the Markov chain is simulated many times to have some statistics (mean, variance) concerning $\tilde{T_0}$, following the principle of Monte Carlo simulation.

5 Framework

5.1 Software Architecture

There are important differences concerning the time scales and tools needed to simulate physics and risk:

1. Physics
   (a) Integration time-step: less than 1 second
   (b) Simulated time: minutes, hours
   (c) Sets of equations
2. Risk
   (a) Integration time-step: day or week
   (b) Simulated time: months or years
   (c) Probability laws, state machine simulation, statistics computation

These differences makes simultaneous simulation impossible: Simulating the physics of the facility over several weeks or months would take too much computation time. Besides, Modelica is not well-suited for the risk simulation,

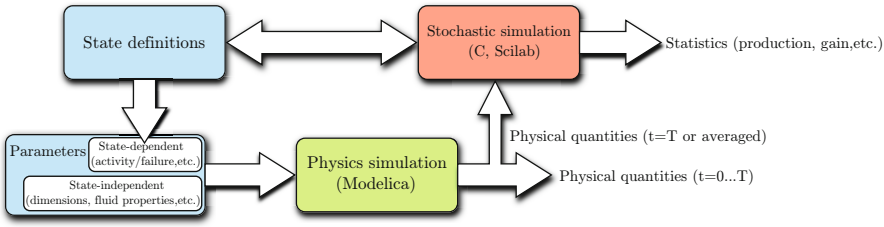


Fig. 6 System-level simulation framework. Inside the risk simulation, the system is a Markov chain. Each state of the chain defines which parts of the facility are working or not. The physics simulation disables/disconnects the corresponding components. The output of the physics simulation then gives the physical quantities of interest related to the simulated state (e.g. production volume).

so a different programming language is required, like C or Scilab[3]. Figure 6 shows the decomposition of the simulation modules.

5.2 Monte Carlo Simulation

Inside the risk simulation, the system is modeled as a Markov chain. The Markov chain is used to simulate virtual life-cycles of the installation, under the randomness hypotheses on the events (price change, failure) that may occur over time. One such virtual cycle corresponds to one realization of randomness. Following the principle of Monte Carlo simulation, many simulations are repeated in order to build empirical distributions of the output of interest (e.g. production or cost). Statistics can be derived from the repeated simulations. From the statistics, indicators can be calculated to obtain information about the system performance. Some indicators concern for instance how the production is affected by the failure events. For instance, resilience[4] aims at quantifying the impact of failures on the production. Other indicators are related to the gain and the risk of loss. Both types of indicator (production and gain) can be applied to one or many simulations. In the latter case, new indicators might be derived to estimate the uncertainty of the performance indicator, e.g. from its estimated variance.

5.3 Multi-level Simulation

A system-level framework must give the user the possibility to select the correct accuracy to answer a specific question, i.e. high enough to answer but not too high so that computation time remains acceptable. For that purpose, different simulation levels must be defined. With Modelica, this is easily achieved through the creation of components that encapsulate different sub-components, each sub-component corresponding to one among the defined simulation levels[3]. Each simulation level has its own equation set and connectors. A global parameter sets the condition to select the same simulation level in all components. The different levels correspond mainly to the different models for the fluid mixture and its flow (compressibility/incompressibility, number of substances, etc.).

6 Conclusion and Perspectives

We have presented a framework for the simulation of an offshore oil facility. The simulator works at the system level, i.e. the general behavior is simulated over a time period that can be as long as the system life-time. The proposed simulation is a combination of physics and risk models. The physical models are based on the equations of physics (fluid flow, heat transfer, etc.) and

[3] `http://www.scilab.org`

engineering (control systems) and on measurement data (phase diagram, hydrate formation diagram, etc.). The risk models are based on stochastic models (Markov chains) and definition of scenarios (price evolution, weather). The final objective is to have a software tool that can be used to freely design and virtually test an offshore installation. This is a very ambitious project, with many ongoing or planned developments. In particular, test cases must be defined to compare the simulated results to real measurements. Starting from relatively simple test cases, the complexity will be progressively increased until industrial scale-one designs. In the meantime, user studies will be conducted to get a feedback from expert offshore installation designers.

References

1. Modelica®- A Unified Object-Oriented Language for Systems Modeling. Language Specification, Version 3.3 (2012)
2. Offshore Reliability Data Handbook. SINTEF Technology and Society (2009)
3. Kuhn, M.R., Otter, M., Raulin, L.: A Multi Level Approach for Aircraft Electrical Systems Design. In: 6th International Modelica Conference (2008)
4. Peck, A., Simonovic, S.P.: Coastal Cities at Risk (CCaR): Generic System Dynamics Simulation Models for Use with City Resilience Simulator. Water Resources Research Report no. 083. Facility for Intelligent Decision Support, Department of Civil and Environmental Engineering, London, Ontario, Canada (2013)
5. Costes, J., Ghidaglia, J.-M., Muguerra, P., Nielsen, K.L., Riou, X., Saut, J.-P., Vayatis, N.: On the Simulation of Offshore Oil Facilities at the System Level. In: 10th International Modelica Conference (2014)
6. Røed, W., Mosleh, A., Vinnem, J.E., Aven, T.: On the use of the hybrid causal logic method in offshore risk analysis. Reliability Engineering & System Safety (2009)

Towards an Extended Interoperability Systemic Approach for Dynamic Manufacturing Networks: Role and Assessment of PLM Standards

Emna Moones, Nicolas Figay, Thomas Vosgien, Lyes Kermad,
François Stephan, Abderrahman El Mhamedi, and El Mouloudi Dafaoui

Abstract. This paper aims at illustrating some limitations of the systemic approach when willing to ensure the interoperability of PLM solutions within a Dynamic Manufacturing Network (DMN), based on e-Business PLM standards and their implementations, being industrial processes, methods, applications or Information & Communication Technologies (ICT) solutions. Indeed, addressing interoperability challenges in such a complex digital business eco-system calls for a holistic approach based on the "system" paradigm. Setting this way, a part of our goal is to underline the limits and drawbacks of such an approach as interoperability brakes and to derive the issues that must be addressed in terms of research in order to remove them. This paper introduces a new approach in order to set up a test bed environment for PLM standards. The required and proposed approach considers a PLM standard not only as a technical solution, but above all as a strategic solution for which it is mandatory to support and enhance discussions between enterprise, product/system, processes, ICT architects and designers. The proposed approach - for analyzing and assessing the relevancy of PLM standards

Emna Moones · Thomas Vosgien · François Stephan
Technological Research Institute SystemX, Palaiseau, France
e-mail: {emna.moones,thomas.vosgien,
francois.stephan}@irt-systemx.fr

Nicolas Figay
Airbus Group Innovations, 12 rue Pasteur 92150 Suresnes, France
e-mail: nicolas.figay@eads.net

Emna Moones · Lyes Kermad · Abderrahman El Mhamedi · El Mouloudi Dafaoui
University Paris8, 140 rue Nouvelle France, 93100 Montreuil, France
e-mail: {l.kermad,a.elmhamedi,e.dafaoui}@iut.univ-paris8.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_5

regarding their usage in specific business contexts - will be illustrated with a multi-layer modeling language. This language is used to model standards-based business collaboration scenarios and to model the test bed environment that will enable the execution/simulation of this scenario and the assessment of related standards implementations regarding the business needs of the scenario. The addressed case study is based on a data exchange scenario between a customer production order scheduler and a supplier production order executer using the ISA 95 standard. From this example, the interoperability issues related to DMN system of systems will be identified, for which accurate test based methods will be defined in future work.

Keywords: System of Systems, Interoperability, Dynamic Manufacturing Network, PLM standards, ISO STEP, ISA 95.

1 Introduction

1.1 Industrial Context

In addition to System Engineering (SE), one trends for manufacturing industry is the application of Product Life cycle Management (PLM). In [1], CIMDATA defines PLM as strategic approach aiming to put in place appropriate processes and solutions for creation and sharing of Product Data and associated processes. PLM applies between enterprises involved in the different phases of the life cycle of the manufactured products and of its components. As PLM solutions relies today systematically on software-based solutions, Product data are digital and have to be interpreted at the same time by the different actors and by technical applications (e.g. Computer Aided Design/Manufacturing solutions) of the numerous enterprises and organizations concerned by the manufactured product. As a consequence, digital e-Business ecosystems are emerging, constituting Dynamic Manufacturing Networks (DMN) for which interoperability of technical applications is a major issue as defined in [2]. In DMN context, interoperability is the ability of the enterprises concerned by a manufactured product to enter the network by interconnecting their private processes, the applications and related technologies supporting these processes in order to ensure secured product and process data exchange and sharing.

In order to respond to digital collaboration needs, numerous industrial groups have been setting up PLM harmonization initiatives (e.g. EADS PHENIX[1]) for which importance of e-Business PLM standards were identified. In such a context, PLM standards are not technical solutions, but strategic solutions that have to be managed consistently by a community of interest (e.g. Aeronautic, Space & Defense European and worldwide community). Importance of selecting and governing a relevant set of open e-Business PLM standards managed in configuration has been identified in different domains, in particular Aeronautic, Space & Defense

[1] `http://www.journeeduplm.fr/uploads/file/eads.pdf`

European community (c.f. ASD SSG [3]). Relying on standards [4] [5] and on models of reference for a community of reference [6] is the only way to achieve continuous and pragmatic interoperability at an acceptable cost. But some barriers remain for achieving such interoperability. In particular the ability for industry to effectively specify how software solution providers must implement standards in order to support their business collaborative processes and in order to facilitate application testing and deployment in industrial operational context.

In addition to the PLM approach and in order to deal with increasing complexity of economic environment, organizations and products, enterprises are also investing more and more in SE. According to INCOSE [7] (International Council of System Engineering), "SE is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem. SE integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation". SE community has been developing its own set of standards, such as ISO 15288 [8], system modeling languages such as SysML [9] or SESTEP application protocol (ISO10303-233) [10].

Some overlapping exists between PLM and SE. According to ISO 15288, the system of interest, i.e. the manufactured product, and the supporting systems, i.e. system for designing, producing, operating and supporting the product, are distinguished. For each of them, all of the phases of the lifecycle are to be considered in order to ensure adequacy between industrial processes and enterprises' capabilities. So stated, it seems that PLM is included inside SE. But the scope of application of PLM is larger than the one covered by SE processes and can be applied being SE processes independent. PLM is also more concerned by the information system and by the technical applications, while SE is more concerned by engineering methods and processes.

Finally, both SE and PLM are concerned by interoperability. While PLM is concerned by data exchange, sharing and long term archiving, SE is concerned not only by possible interaction between systems and by automated reconfiguration of system of systems (SOS) but also by enhancing communication and hence interoperability between multi-disciplinary design teams. Moreover, SE also focuses on the adaptation of the overall system in order to respond to the targeted objectives and on the way the different sub-systems of a SOS have to be aggregated dynamically and to interact easily.

1.2 Research Context and Orientation of the Proposal

The research work presented in this paper is related to the research project "Standards Interoperability PLM" (SIP[2]) launched within the frame of the

[2] http://www.irt-systemx.fr/systemx-lance-le-projet-sip-standards-interoperabilite-plm/

IRT-SystemX. As defined in [5], this project has three main objectives. First objective is the development of a generic approach and framework for specifying and testing implementation of PLM standards for multi-disciplines and multi-sectors collaboration. Second objective is the promotion of an experimental capability research for developing, assessing and implementing a configured set of relevant PLM standards which covers the whole phases of the life cycle of an industrial product. Third objective is the enhancement of the PLM Interoperability maturity of industry: for any stakeholder or actor of the domain, it is allowed accessing, assessing and contributing to the results of the project. The goal is to create a sustainable (i.e. which will continue to exist after the end of the project) open community which will be able to drive development by software product providers of accurate PLM solutions with validated specifications and ability to test them within a DMN.

Because PLM and SE are closely related, the project has to consider standards and practices of both PLM and SE communities. The project also develops a global interoperability approach that will extend SOSI approach taking into account virtualization aspects. Our goal in this paper is limited to point out an interoperability brake which complete the set of brakes defined in [11], and related to some limitation of system paradigm for DMN Interoperability.

The section 2 of the paper will describe the research foundation of the SIP approach. The section 3 will describe and analyze the limitations of systemic for addressing interoperability. The section 4 will illustrate the SIP approach and the previously described limitations for a case study related to the ISA 95[12] standard, and to enterprise control integration. Conclusion will introduce perspectives and future work.

2 SIP Related Work

2.1 Positioning According Interoperability State of the Art

The SIP approach is closely related to [11], in which the author proposes a federated framework for interoperability of technical enterprise applications. This framework first states what is an enterprise application and what is interoperability of enterprise applications. It then qualifies the “ideal” information system for networked collaborative product development. On the basis of past research projects and operational projects, the author also analyzes why PLM standards are not used, identifying set of interoperability brakes (i.e. what lead to non-interoperability) and interoperability enablers. Enablers include those defined in the ATHENA project [13], which considers that interoperability must be addressed at different layers - business, knowledge and ICT (Information and Communication Technologies) - with inter-related ontological models defined at each layer. In addition, a model driven approach is used in order to “project” the business logic (business objects, services and processes) on a service oriented execution platform including service bus, application servers and workflow engines.

Complementary proposed enablers are the systematic usage of open standards, the need for preparing and constructing operational interoperability as defined by SOSI. Another important identified enabler is the establishment of a community of interest to build its maturity through the governance of a consistent set of standards covering their needs. The brakes can be considered as practices adopted by enterprises which are going against interoperability. An example is the management by project. As a project has restricted duration and scope, long term and global strategic approach at enterprise scale are usually not considered. As a consequence, using a neutral standard for interchange is considered at the project scale as an important extra cost, and is often not considered as a first priority. Needs for management, evolution and consistency of the whole enterprise information system are not considered.

Alternating research projects and operational projects with continuous update of enablers and brakes is another principle of the framework. Doing so, the framework has been completed through Crescendo [14] project for integration of an enterprise portal as part of the execution platform, in particular in order to deal with controlled access to resources of the enterprises. The brakes addressed here is security, which is a stopper when not achieved. Standards for simulation were considered such as ISO10303-209 [15] and CGNS [16]). The IMAGINE project[3] has been addressing dynamic allocation of actual qualified resources to a process within a DMN. Concerned applicative resources might implement PLM standards in order to support seamless information flow all along cross-organizational collaborative project. In addition, usage of virtualization servers has been adopted in order to facilitate deployment, set up, and simulation of an actual DMN over the public or private clouds. Finally, ArchiMate has been adopted as the open standard to be considered for enterprise modeling promoted in ATHENA. ISA95 and ISO15288 were assessed and combined. The brake addressed by IMAGINE is the lack of methodology for qualification of a set of applications involved in a cross organizational collaborative process.

2.2 Positioning According Test Beds State of the Art

SIP was built on top of the results of these successive projects, in order to address brakes related to missing methodology for producing use cases, business scenarios, test data sets and test procedure, positively impacting implementation costs for making solution providers implement the standards. Referring to existing test beds such as NIST QOD and CVTS[4], Korean B2B interoperability test bed, Global eBusiness Interoperability test beds (GITB) or Agile Test Framework (ATF), it appears that none of them is addressing the need to consider implementation of standards by the engineering processes first before to specify implementation of interfaces within commercial solutions.

[3] http://www.imagine-futurefactory.eu/index.dlg/
[4] http://www.nist.gov/el/msid/qod_standalone_release.cfm/

The innovative aspect of the SIP project concerning test beds is also the ability to consider several standards in a holistic way, with combined usage of Business standards (e.g. ISO 15288 technical processes), applicative standards (e.g. application data interchange protocols such as ISA 95 or ISO STEP) and ICT standards for data exchange (e.g. XML [17]), distributed services (e.g. WSDL [18]) or process choreography (e.g. XPDL [19]). All these standards are mapped within enterprise models formalized with ArchiMate, allowing enterprise, business process, product, information system and ICT architects to establish PLM interoperability through industrialization of standards.

2.3 SIP and System Engineering

SIP is closely related to SE by several aspects. First the SE process framework defined by ISO 15288 is used for contextualization of PLM standards, but also for making a clear distinction between the system of interest (e.g. an aircraft) and supporting systems (i.e. system for designing, system for producing, system for operating or system for supporting). Then the SysML standard is one of the PLM standards considered for the support of some of the technical processes related to requirement engineering, design and simulation. In addition, usage of model driven approach for referential component relies on the Unified Model Language, which support both object and component paradigms. An object groups data and methods for systems which have to interact through exchange of messages. It considers what is internal to objects and what is external. Internal part can be accessed through public interfaces. Object classes are used for categorization of objects, and support inheritance mechanism in order to ensure reusability. Such mechanism brings an important drawback, due to the complexity of inheritance trees and usage of specialization/generalization for combining business, applicative and technical objects. It led to the failure of standards such as PDM Enablers, too difficult to implement as it required mastering business and ICT specifications. The “component” paradigm provides the concept of container: a container is provided by an application server and allows deploying business objects. ICT services provided by an application server can be applied to the business through the containers, according to policies applied to these containers. Doing so, business and ICT aspects are decoupled, allowing separation of concern. Business logic can be deployed as engineering artefacts on top of execution platform. It is so possible to execute business logic.

Finally, when considering DMNs, it is required to interconnect legacy applications supporting organization which have to collaborate dynamically, with continuous evolution of organizations, processes and ICT leading to dynamic reconfiguration if willing to ensure continuous interoperability. It seems that DMNs can be considered as SOS, and consequently it should be possible to use methods and tools defined by the SOS community. It was done with SOSI principles, which define operational interoperability which has to be prepared by mean of governance (c.f. ASD SSG) and constructed by mean of architectural patterns.

But some issues exist when using the "system" paradigm. It can be analyzed comparing our global interoperability approach (taking into account virtualization aspects) used in SIP and other systemic approaches like SOSI.

2.4 Architecture and Principles of SIP Test Bed

The SIP test bed (c.f. Figure 1), includes first an execution platform combining standardized collaborative portal, workflow engine and enterprise service bus. On this execution platform, testing and standard based PLM services are deployed while a shared repository of use cases, test scenarios and test data set. Infrastructure and processes of the SIP test bed make possible controlled access to the services and the repository, which can be public or restricted to a given community or enterprise.

The SIP methodology allows generating referential implementations of applicative components from models (based on MDA [20]) that will simulate the different kind of applications (type A or B in the figure) that will be interconnected for supporting the collaboration.

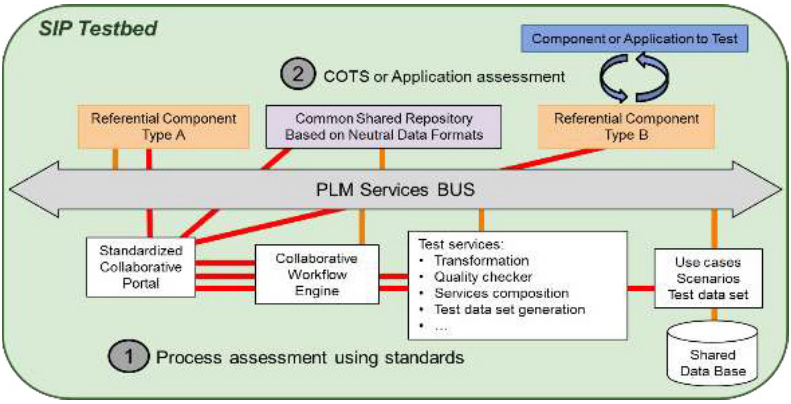


Fig. 1 SIP test bed architecture and principles

Once these components generated, deployed and interconnected, it will then be possible to simulate standards-based collaboration and then to assess as well the used standards implementations (are they covering business needs?) than the cross organizational collaborative processes of a given digital business ecosystem such as Technical data package exchanges, Change and Configuration process management, etc. Once validated by the mean of the test bed, enterprises will then be able to precisely specify to software solution providers and to integrators what is needed in order to be able to interconnect actual applications and their interfaces. When actual applications and their interfaces ready, it will then be possible to test them reusing the test bed: referential components will be unplugged, and replaced by the actual application. By playing the same test scenarios and reusing the same

test data, it will be possible to assess and qualify actual applicative components, performing first unitary tests (one component alone) and integration tests for a whole end to end process involving at the same time different organizations, different applications and potentially a set of different PLM standards.

SIP will not perform all the work, but invite partners and communities to apply SIP methodology and to use and enrich the SIP platform with new open components in order to build maturity of the industry concerning PLM interoperability, being for design, production or integrated logistic support.

3 Systemic and Its Limitations for a Global Interoperable PLM Approach

The theory of systems was founded by Ludwig von Bertalanffy, William Ross Ashby and others between 1940 and 1970. It evolved as the study of the complexity, with a particular focus on dynamic and evolutionary systems. Systemic analysis is an inter-disciplinary field related to the study of complex objects which can be understood with classical approaches (e.g. Cartesian method), such as living being or electronic systems for temperature regulation. In order to face such a problem, it is required to adopt a holistic approach, focusing more on the exchange and interactions (interaction, retroaction, regulation) between the different parts of a system than on the analysis of each part, and considering the objective of the system (teleology). Systemic approach is applied to numerous domains: biology, architecture, etc. It relies on visual modeling, descriptive or formal, executable or not. With executable models, it is possible to use simulation. As stated by AFSCET [21], the problem of boundaries is a key when willing to deal with what is internal and external and to be able to define the interactions between the systems.

Looking at the complexity of a DMN, it seems that systemic approach could be appropriate for addressing PLM interoperability. As for systemic approach, the SIP interoperability approach is holistic, and considers different systems: the system of interest (i.e. the product), the supporting systems, the information system, the enterprises, the digital business eco-systems, etc. Nevertheless we identified some difficulties concerning usage of systemic approach in the PLM interoperability context, when willing to define boundaries for a system. The origin of this difficulty is the virtualization.

Virtualization refers to the act of creating a virtual (rather than actual) version of something. This has been used since a long time in computer science, with as an example usage of logical disk names in order to be able to change the actual used physical disk in a transparent way without impacting the users. Virtualization has many other usage and applications to be considered in SIP. Enterprise portals are software systems which aim to give users access to numerous applications through an integrated interface, hiding the complexity of the underlying actual architecture of solutions realizing these applications. It is true in particular for

PLM hubs used by a digital business eco-system. In order to easily deal with a simulation platform, SIP is making intensive usage of virtualization servers in order to reduce time of deployment and replication of a whole collaborative network. Cloud computing and Grid computing are making extensive use of virtualization. Finally, enterprise modeling and associated standards are interconnecting the enterprise and the actual ICT system using business layer, applicative layer and ICT layer. The ICT layer is constituted of concrete devices and software systems used in order to realize an application. The applicative layer is purely logical, and makes the interface between ICT technologies and the business. ICT devices and software systems are concrete actual systems that are owned by an organization, which have physical location and which are physically operated through accurate processes by organizations. Owner of the ICT capabilities, ICT capabilities and operators of the ICT capabilities can as well be inside or outside the enterprise using the application.

As a consequence, as soon as virtualization is used, it is not possible to preserve boundaries of a system between Business and ICT layers. When interactions exist between two organizations, there are not necessarily interactions between two software systems installed on different machines. Conversely, one organization can access one application without knowing it implies interaction between numerous software systems distributed on different machines and eventually hosted within numerous organizations. Considering grid computing which allocate dynamically available actual resources to an application, it is impossible to predict what will be the actual used resources and where they will be located.

4 Illustration through ISA95 Case Study

4.1 ISA 95 Standard for Enterprise Control Integration

ISA-95 is an international multi-part set of standards that defines interfaces between enterprise activities and control activities. Developed for global manufacturers, it applies in all industries and in all sorts of processes, like batch processes, continuous and repetitive processes. Four functional levels are defined by ISA 95 standard. Levels 0, 1 and 2 are the levels of process control. Their objective is the control of equipment, in order to execute production processes that end in one or more products. Level 3 could be called the level of MES (manufacturing execution system) activities, it consists of several activities that must be executed to prepare, monitor and complete the production process that is executed at level 0, 1 and 2. The highest level (level 4) could be called the level of enterprise, including ERP (Enterprise Resource Planning) systems and PDM Systems. At this level financial and logistic activities are executed in order to produce the product configuration ordered by the client. ISA 95 focus on the Production system, which is a supporting system according ISO 15288. The system of interest (ISO 15288) is the product, which is the output of production activities (ISA 95). The ways for

describing Product data are very different between Production departments (ISA 95, B2MML), design offices (ISO STEP AP242) or customer support department (ISO STEP ISO AP239), as the purpose and the goal of their activities are not the same. A PLM approach should address consistent usage of this set of standards.

4.2 *Modelling and Simulation of a DMN Collaboration Process*

On Figure 2, the different applications are realized through simulators hosted on the test bed (referential components) and virtualized on the cloud. As a consequence, the physical realization of the applications is out of the enterprise systems boundaries. When real application is ready to be tested, the referential component is unplugged and replaced by the real application. It is fully transparent at business layer where used solutions realizing the applications are hosted.

The Figure 3[5] is an illustration of captured processes covered by ISA 95, underlying information system and ICT layer, including the SIP test bed infrastructure.

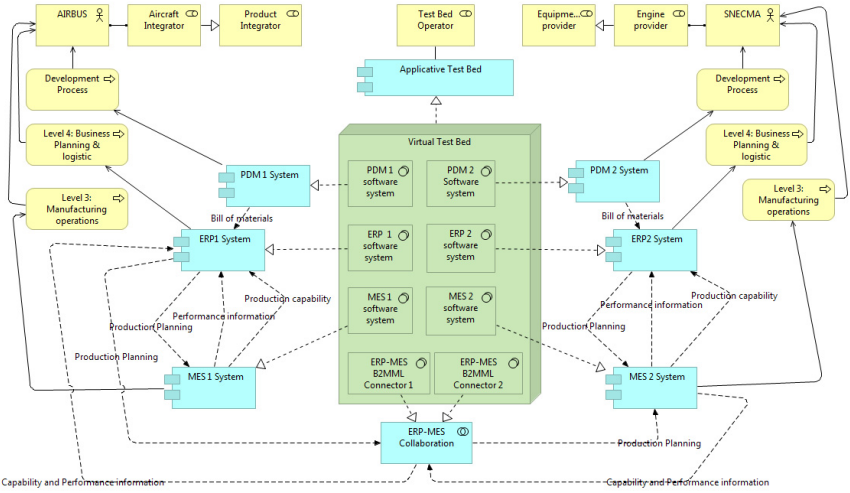


Fig. 2 ArchiMate view of SIP test bed for ERP/MES/PLM solutions

The upper part is the collaboration scenario model, which involves a "customer order scheduler" and a "supplier order executer" (grey). The order scheduler wants to transmit to its supplier a production schedule as a set of production requests (or orders) with associated required manufacturing bill of materials respectively from

[5] http://www.eads-iw.net/image/image_gallery?img_id=162879&t=1404673922104/

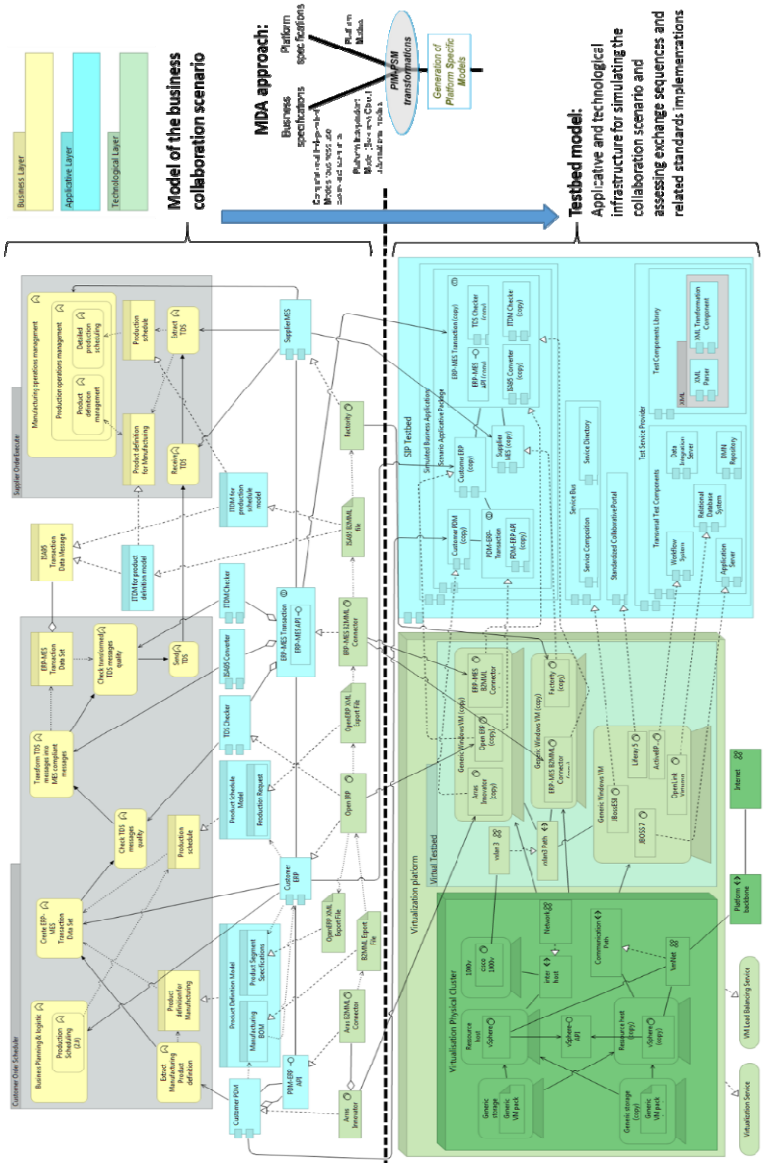


Fig. 3 ArchiMate view of ISA95 business case on top of SIP Test bed

its ERP and PDM systems. The supplier might be able to integrate all these information into its MES system in order to define its detailed production planning. This scenario is modeled on the three layers. The business layer (yellow) specifies the sequence of business activities and related business objects. The applicative layer (blue) specifies the applicative components supporting these activities and the data objects realizing the business objects. The technological layer (green) specifies the actual software systems and the data files respectively realizing the applicative components and data objects.

The lower part represents the test bed model; i.e. the applicative (blue boxes on the right side) and technological infrastructure (green boxes on the left side) for simulating the collaboration scenario and assessing exchange sequences and related ISA95 implementations. In the technological layer (the virtualization platform), we modeled and distinguished the virtual physical cluster specifying the physical and "real" infrastructure nodes describing a set of virtual machines and the virtual test bed composed of these latter. These virtual machines host the simulated applications/implementations, and the physical storage devices host the generated or consumed test data sets but also enabled the workflow models of the test scenarios. Figure 3 highlights not only the complexity of modelling such a collaboration scenario on the three layers, but also the difficulty to define system boundaries that can be preserved between the business, applicative and ICT layers of a complex DMN model.

5 Conclusion and Way Forward

In this paper, we have illustrated some limitations of the systemic approach when willing to ensure the interoperability of PLM solutions within a DMN. It was done within the context of our research activities related to PLM interoperability based on standards. We aims to propose a methodology, which was just introduced in this paper, for dealing with DMN, based on a holistic approach derived from the federated interoperability framework, and addressing the interoperability brake related to missing adapted approach for use cases and test scenarios. The methodology, relying on a test bed allowing execution and simulation of DMN models, and the approaches developed for system of systems, are very similar: high complexity, iterative usage of modeling and simulation. We also reuse SOSI concepts, for preparing and constructing operational interoperability, and we rely on System Engineering process framework for contextualization of PLM standards regarding System Engineering process standards. But due to virtualization, it is not possible to define system boundaries that can be preserved between the business, applicative and ICT layers of our DMN models. We have illustrated some limitations of the systemic approach and system paradigm by modeling a business case related to the usage of the ISA 95 standard, for inclusion of the production function in an interoperable PLM approach. The methodology we are developing will address such limitations, but also other interoperability brakes we identified and that will be described in future papers. We recommend also considering the lifecycle

impact of design/support tools for software components in the system of interest and what advantages could the use of web based interoperability technologies provide. Our approach will apply to manufacturing standards for production systems, but also to design systems (Computer Aided Design, Configuration Management and Simulation) and integrated logistics systems.

Acknowledgement. This research work has been carried out under the leadership of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program "Investissements d'avenir".

References

1. CIMDATA: All about PLM (2014), https://www.cimdata.com/en/resources/about-plm
2. Figay, N., Tchoffa, D., Ghodous, P., Exposito, E., El Mhamedi, A.: Dynamic Manufacturing Network, PLM Hub and Business Standards Testbed. In: Enterprise Interoperability VI, pp. 453–463. Springer International Publishing (2014)
3. ASD SSG: ePLM Interoperability (2014), http://www.asd-ssg.org/
4. Lianga, J., Shaha, J.J., D'Souzaa, R., Urbanb, S.D., Ayyaswamyb, K., Harterc, E., Bluhmc, T.: Synthesis of consolidated data schema for engineering analysis from multiple STEP application protocols. Computer-Aided Design 31 (1999)
5. Zha, X.F., Du, H.: A PDES/STEP-based model and system for concurrent integrated design and assembly planning. Computer-Aided Design 34 (2002)
6. ATHENA: Interoperability Framework v2.0 (2007), http://www.nehta.gov.au/implementation-resources/ehealth-foundations/EP-1144-2007/
7. INCOSE: International Council on Systems Engineering (2014), http://www.incose.org/
8. International Organization for Standardization: ISO 15288:2008 - Systems and software engineering - System life cycle processes (2008)
9. Object Management Group: SysML Version 1.3 (2012), http://www.omg.org/spec/SysML/1.3/
10. International Organization for Standardization: ISO 10303-233:2012 - Industrial automation systems and integration - Product data representation and exchange - Part 233: Application protocol: Systems engineering (2012)
11. Figay, N.: Interoperability of technical enterprise applications, Doctoral Thesis, University Lyon 1 (2009)
12. ISA-95: the international standard for the integration of enterprise and control systems (2014), http://www.isa-95.com/
13. Athena Interoperability Framework, AIF (2010), http://athena.modelbased.net/
14. CRESCENDO: Collaborative and Robust Engineering using Simulation Capability Enabling Next Design Optimization (2014), http://www.crescendo-fp7.eu/
15. International Organization for Standardization: ISO 10303-209:2001 - Industrial automation systems and integration - Product data representation and exchange - Part 209: Application protocol: Composite and metallic structural analysis and related design (2001)

16. CGNS, NASA, CFD: General Notation System, Version 3.2.1 (2014), http://www.grc.nasa.gov/WWW/cgns/CGNS_docs_current/
17. W3C: Extensible Markup Language (XML), Version 1.0 (2013), http://www.w3.org/TR/xml/
18. W3C: Web Service Definition Language (WSDL), Version 1.1 (2001), http://www.w3.org/TR/wsdl/
19. WFMC: XML Process Definition Language, XPDL (2014), http://www.xpdl.org/
20. OMG: MDA – The Architecture of Choice for a Changing World (2014), http://www.omg.org/mda/
21. Donnadieu, G., Durand, D., Neel, D., Nunez, E., Saint-Paul, L.: L'approche systémique: de quoi s'agit-il, Synthèse des travaux du Groupe AFSCET « Diffusion de la pensée systémique » (2010), http://www.afscet.asso.fr/SystemicApproach.pdf

Flexible Queries over Engineering Data

Yishai A. Feldman

Abstract. The design of complex systems involves many engineering disciplines, and many different tools and formalisms. Solutions such as IBM Rational Engineering Lifecycle Management(RELM) present a unified view of information collected from multiple tools. Various queries and reports are predefined, but engineers need to define their own custom queries, based on their specific processes or products.

This paper presents a classification of various types of useful queries and rules over the combined information. Some rules could not be expressed precisely, because of missing information in the repositories. Other rules had to be stated at a lower level of abstraction than desired, because of the level of the concepts in the repository. The paper gives examples and suggests ways to remediate these issues.

A visual query formalism based on SysML object diagrams to express these queries is described. The visual query language has been implemented and used to express rules and find violations in two repositories.

1 Introduction

Complex systems engineering involves multiple disciplines, including mechanical, electrical, control, and software. Each discipline has its set of tools; these are often not integrated with each other. In some cases, there is ad-hoc integration between a pair of tools; for example, organizations commonly create connections using ad-hoc scripts. However, this is expensive, brittle, and

Yishai A. Feldman
IBM Research, Israel, Haifa
e-mail: yishai@il.ibm.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_6

error-prone. The Open Services for Lifecycle Collaboration (OSLC) set of standards [1] specify a common vocabulary for various domains related to software and systems engineering, including change management, requirements management, and quality management. Solutions such as IBM Rational Engineering Lifecycle Management® (RELM) [2, 3] are based on OSLC, and present a unified view of information collected from multiple tools. RELM contains various predefined queries and reports, but these are insufficient; engineers need to define their own custom queries, based on their specific processes or products. Because RELM is based on semantic-web standards [4], queries over its contents are expressed using the SPARQL query language [5]. However, engineers who are not software developers cannot be expected to express their queries in SPARQL. They would need a different formalism, which is familiar and more natural for them to use.

This paper examines the following research questions:

- What classes of valuable queries and rules is it possible to formulate based on the collected information? Section 2 gives a classification of queries with examples for each, including queries related to both process and systems.
- What are the difficulties of expressing these queries and rules in a general way? Section 3 describes required information that is missing from the repository, and various abstractions that need to be provided to support a flexible query mechanism.
- How can such queries be expressed in a form that would be usable by systems engineers and other end-users? Section 4 suggests a visual formalism based on UML/SysML object diagrams for representing queries and abstractions. The semantics of the formalism is defined in Section 5, and results of experiments with the implementation are described in Section 6.

1.1 Technology Background

RELM is built on top of the Jazz Foundation® [6], which is a scalable, extensible, team-collaboration platform that integrates tasks across the software lifecycle. Jazz is built according to the OSLC standards, and stores information as subject-predicate-object triples in the Resource Description Framework (RDF) format [4]. Triple elements that represent external objects are called *resources* in RDF, and are expressed as Uniform Resource Identifiers (URIs).

The semantics of some of the predicates and values used in Jazz and RELM is specified in OSLC. However, the standards do not specify everything, and much is left to the tool implementer's discretion. See Section 2.1 for examples.

The standard query language for semantic-web data is SPARQL [5], which is an adaptation of SQL to RDF triples instead of relational-database tables. It is similar in complexity to SQL, and is meant for experts rather than casual users. It is therefore inappropriate for use by engineers who would only need to specify queries on an occasional basis.

1.2 Use Cases for Queries

Queries can be used for many different purposes. A systems engineer or process owner may want to express validity rules over the data in the repository; violations of these may need to be fixed, if possible with tool help. Managers at various levels may want to express queries that provide metrics about the quality of the process and product. In particular, they could view rules as queries that provide compliance metrics. Some rules express best practices that may be overridden in certain cases. The line between rules and queries is therefore quite fuzzy.

Many interesting rules refer to information from multiple sources. Such rules can function as preconditions on state transitions, preventing illegal transitions. When that is not possible or practical, rules may be associated with remediation actions.

Queries can also function as sanity checks, to prevent common types of errors. The following real stories are typical:

> I needed 5 screws for a component, but when the parts arrived it turned out that the units were boxes, so I received 50,000 screws instead.

In this case, a rough estimate of cost may or may not have revealed the mistake, but an estimate of weight may well have triggered an alert (see Section 2.5).

> In the shipment of the parts for a communications system we received a large box containing a truck wheel. We protested and tried to send it back, but it turned out that the catalog number for the power supply was wrong.

A classification of parts and a set of rules of what goes together might have prevented this mistake (see Section 2.5).

From one important point of view the (systems or software) development process can be considered as a large number of artifacts, each going through a set of state changes. These state changes have mutual dependencies and constraints, and are often tracked using different systems. For example, a bug report can cause the creation of a defect, which can cause the creation of several work items. The states of the bug report, defect, and work items are obviously related, and inconsistent mutual states indicate a process error.

There are many constraints on all kinds of models used in systems and software engineering; these can be expressed using rules on the structure of the models; see Section 2.6.

The results of these rules and queries can be presented in various ways, depending on the viewpoint of the requester. Engineers will need access to the list of detailed results, to check which indicate violations that need to be fixed. Managers may require statistical and historical data in order to obtain insight into the state of the development process. This paper focuses on the expression and computation of the queries; further processing of the results can be done by downstream tools.

2 Queries and Rules on Engineering Data

This section presents a number of different types of rules, with several examples of each. The list is not exhaustive, but seems to represent a useful and diverse set of needs.

2.1 Existence Rules

The following rule requires the existence of some resource based on other resources found in the repository:
R1 Every work item of type Story, Defect, Task, Change Request, Impediment, or Enhancement must have at least one associated test case.
R2 Every unit test must be associated with at least one work item.
R3 Every requirement must have at least one artifact that implements it.
R4 In a completed design for a component, each atomic part must have exactly one associated catalog item.

All the existence rules seem to share a general pattern, and this is true for other classes of rules as well (see below). This indicates that query patterns should be supported by a user-friendly tool; see Section 4 for further discussion.

2.2 Consistency Rules

The following rules express constraints on the development process:
R5 A work item cannot be in completed state unless all tests for it have passed.
R6 A component cannot be declared "ready for assembly" unless all parts are associated with specific catalog items in the bill of materials (BOM).

Consistency rules relate the state of one item with the states of other items. As explained in Section 1, such rules are relevant whenever there is an interaction between states of different items, which is very common.

2.3 Ontological Consistency Rules

Ontological consistency rules check the repository for illegal or unlikely situations. For example, the property "tested by test case" relates a work item to a test case that is supposed to test its contents. The property "tests work item" relates a test case to the work item it tests. It seems that these properties should be inverses of each other:
R7 The properties "tested by test case" and "tests work item" are inverses of each other.

This means that if one link exists between two objects, the other should exist between the same two objects in the other direction.

Note that this is not stated this way in the OSLC standard. In fact, the standard says this about the `testedByTestCase` property: "It is likely that the target resource will be a `TestCase` but that is not necessarily the case." In fact, the implementation described in Section 6 found violations of this rule, although these seem to be bugs in the repository data.

This is a very general pattern, applicable, among other things, to many structural relationships. For example:

R8 The properties "part of" and "contains part" are inverses.

2.4 Non-circularity Rules

Change management tools such as IBM Rational Team Concert (RTC) define a blocking relationship between work items. Obviously, circular blocking dependencies must not exist, since they represent deadlocks. It turns out, however, that it is easy to generate such cycles in RTC.

R9 There must not be a cycle in the "blocks" relationships between work items.

This is also a very general pattern; another example is:

R10 There must not be a cycle in the "part of" relationship.

Rules of this class check for structural errors in the process or the product.

2.5 Domain-Specific Rules

The first quote in Section 1, regarding the error in the number of screws ordered, is a symptom of a common problem. Mistakes in units or in numbers (e.g., missing or extra zeros) are easy to make and difficult to catch. Sanity-checking rules such as the following are easy to state:

R11 A family sedan must have exactly 4 wheels.

R12 A desktop computer can have no more than 10 extension cards.

These examples deal with quantities; the second quote in Section 1, regarding the error in the item ordered, is a type of consistency issue. Examples of sanity-checking rules of this class include:

R13 All tires of a car must be of the same type.

R14 An aircraft must not have any non-airborne components (such as sonar).

R15 Products to be exported must not include components subject to export limitations (such as ITAR).

This is also a very general class of rules; consistency can be defined between items, as in **R13**; between components and subcomponents, as in **R14**; or between external properties of items stemming from regulations, policies, or other criteria, as in **R15**.

Given a large enough repository, such rules can be learned from the data itself. However, such repositories do not currently exist, and so this line of research is left for the future.

2.6 Modeling Rules

An increasingly important aspect of systems engineers' work is the construction of various kinds of models. There are many constraints on models, such as the following:
R16 If two variables have the same stereotype, their types (e.g., integer/real) must also match.
R17 The cost for a catalog item cannot be missing or zero.
R18 If a SysML block has the stereotype «Technical», all contained blocks must also have the same stereotype.
R19 The same block may not simultaneously have the stereotypes «Technical» and «Geometrical».
The last two rules are examples of very general patterns: transitivity according to structure (**R18**), and mutually-exclusive properties (**R19**).

Another example of a structural property is:
R20 The types of connected ports should match.
This becomes more complicated if the connected blocks belong to different tools; for example, if a Rhapsody block is exported as a Functional Mockup Unit (FMU) [7] and used in the context of a different tool that has a different type system, the definition of which types match in the two tools becomes more complex.

This seems to be a very rich area for many types of rules. Some of these apply to a single tool and are general enough that they can be embodied in the tool itself. Others, however, cross tool boundaries or are project-specific. These require the flexibility and expressive power to allow customers to create and manage their own rules.

3 Requirements from Flexible Queries

The set of types of work items that should have associated test case in rule **R1** depends on the process, and can be configured for each project separately. It follows that this rule cannot be pre-configured with a change-management tool. It might be possible to create a generic rule that can be parameterized with the set of relevant types; while specific examples can be generalized in this way, it seems that the range of existence rules is too large to be covered by a relatively small set of parameterized rules. A convenient way of expressing rules is therefore necessary for use by engineers.

It is crucial that systems engineers and other end-users not have to specify the precise types of RDF resources or the relationships between them. For example, an OSLC work item (called a "change request") is a resource of type `http://open-services.net/ns/cm#ChangeRequest`, and the relationship between a work item and a test case that is relevant for rules **R1** and **R2** is `http://open-services.net/ns/cm#testedByTestCase`. Users will not be able to specify these resource strings, and should not even be exposed to them.

This means that a flexible library of concepts (including resources and relationships) needs to be supplied; it should contain abbreviations and defaults for types and relationships.

Similarly, the property used by the blocking rule (**R9**) is not specified by the OSLC standard; RELM uses the vendor-specific resource `http://jazz.net/xmlns/prod/jazz/rtc/cm/1.0/com.ibm.team.workitem.linktype.blocksworkitem.blocks`, which reinforces the need for abstraction.

In order to hide these details from end users, we use a library that contains a list of all known types. Each type is annotated with an abbreviated name, such as "WorkItem" or "TestCase". These names may or may not be related to the names used in the repository; their purpose is to be intuitively clear to end users. Each type is also associated with its attributes, and with the associations it has with other types. Because the standards are vague on range types (see Section 2.3), this is not well defined. However, given one or more repositories, it is possible to mine them for the actual relationships that they contain. The implementation described in Section 6 contains a utility that creates an initial library based on information in existing repositories, with simple heuristics for abbreviations. This initial library can then be modified manually if necessary.

Rule **R5** shows the need for another kind of abstraction. While the concept of a passing test case is natural from the point of view of a process owner, test cases in OSLC do not have states signifying success or failure. (Test cases do have their own states; for example, those associated with the formal review process.) Instead, test cases have associated test-case-execution records, and these carry the pass/fail status. For a test to be considered to have passed, the latest test-case-execution record associated with it (i.e., the one with the largest creation time) must be in the "passed" state. This example demonstrates the need for creating abstract states of objects, which in reality are based on the states of other objects to which they are related. Section 4 suggests a visual formalism for expressing such abstract states.

Similarly, there is no definition of what it means for a work item to be in a completed state. In a Scrum process, this can be interpreted as one of the states "Done", "Approved", "Closed", or "Resolved". However, other processes can define this concept in other ways, and the specific strings are vendor-specific, as they are not specified in the OSLC CM standard. Furthermore, these are customizable by clients, so the meaning of a "completed work item" can change by organization, team, or project, again demonstrating the need for a flexible and hierarchical abstraction mechanism.

Rule **R5** raises a number of questions about the ways rules are triggered and the responses to violations found. One interpretation of this rule is as a precondition on a state transition:

> A work item cannot move to a completed state unless all tests for it have passed.

This can only be enforced if the rule is triggered by the attempt to change state, either by the tool responsible for that state, or by that tool checking with some other server. If rules are run periodically on a separate server, this interpretation is not viable; instead, it is necessary to perform some remediation action. The simplest, and easiest to automate, is just to create a work item requiring the violation to be resolved. A better option, which is specific to this rule, may be to re-open the work item that has a failing test.

The following interpretation of the same rule is exactly analogous to the previous one:

> A test case cannot move to a failed state if the corresponding work item is closed.

This is obviously wrong, but the reason it is wrong has to do with information about the process, which is outside the rule itself. This demonstrates that rules should be accompanied by the specification of possible actions, including when they can be used as preconditions, and what remediation actions are possible.

The concept of a unit test, which appears in rule **R2**, presents another issue. Some tests, such as acceptance tests, are not associated with work items (they could be associated with requirements, for example). However, the OSLC standard does not distinguish between unit tests and acceptance tests; both are just pieces of code. Without this information, rules such as **R2** cannot even be formulated. Similarly, there is no formal distinction between production and testing code.

A rule that functions as a precondition on a state transition may need to know the target state of the transition; and if the same rule is used to check validity after the transition has already taken place, it may need to know the previous state. This information is not currently available in the repository, although it could easily be obtained and stored.

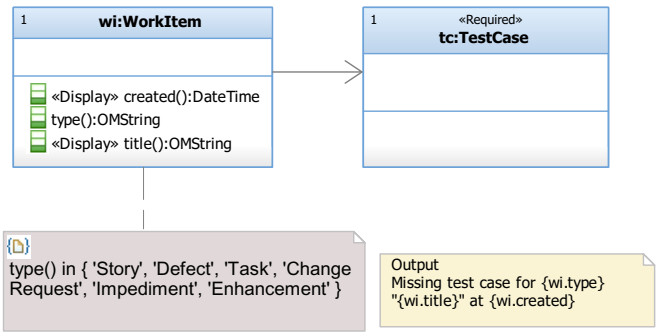


Fig. 1 Visual notation for Rule **R1**

4 A Visual Query Formalism

As mentioned above, it is unreasonable to expect engineers to use SPARQL to formulate queries or rules over the combined engineering data. Because so many queries are structural in nature, it is natural to create a notation that is based on an existing standard for expressing structural properties of objects, namely, UML/SysML object diagrams. These represent each object as a box containing its name and type, and any attributes or methods it may have. For the purpose of the query language, there is no difference between fields and parameterless methods, and either can be used (we will use the term *attributes* in the sequel). Relationships between objects are represented using various kinds of edges; the query language uses only directional associations. The diagrams are extended with a profile that includes several stereotypes to express the semantics of the query or rule.

For example, Figure 1 shows the visual notation for rule **R1**. It contains two objects, a `WorkItem` named `wi`, and a `Testcase` named `tc`. The work item contains three attributes: `created`, `type`, and `title`. The associated constraint (below the object) requires the type to be one of the six mentioned in the rule. (The constraint language used is a simpler variant of UML's Object Constraint Language [8].) The «Required» stereotype on the test case means that this object must exist according to this rule; in other words, the rule is violated if it does not exist. The «Display» stereotypes on two of the attributes of the work item mean that these should be displayed for every violation found. However, in this case there is also an explicit output specification (lower right), which provides a more readable text.

In accordance with the discussion of Section 3, the notation of Figure 1 makes several abstractions over the RDF representation of the data in the repository. First, types are abbreviated; for example, `WorkItem` is used instead of `http://open-services.net/ns/cm#ChangeRequest`. Also, the association between the work item and the test case is unnamed, defaulting to `http://open-services.net/ns/cm#testedByTestCase`. These abstractions are supported by the library (see Section 3).

Figure 2 shows the visual notation for rule **R5**. In this case, the «Required» stereotype is attached to the constraint on the bottom right, specifying the passed status of the test result. The test result itself is the latest one, as indicated by the «Largest» stereotype on its `createdAt` attribute. This form of the rule exhibits the type and association abstractions as the previous one, but still refers to the explicit list of work-item states considered to be "completed," and to the vendor-specific test-result status indicating that the test passed. Ideally, we would like to represent this rule as shown in Figure 3, where the «Local» stereotype on the `status` attribute of the test case indicates that this is not found in the repository, but instead is inferred by another rule, which would use the «Infer» stereotype on that attribute (not shown for lack of space). Another rule (not shown) would abstract the status

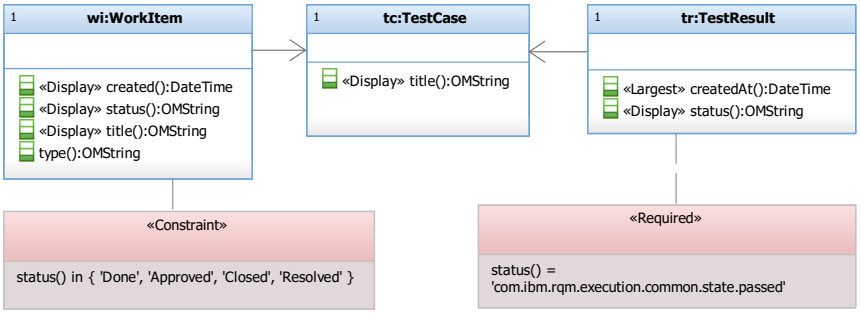


Fig. 2 Visual notation for Rule **R5**

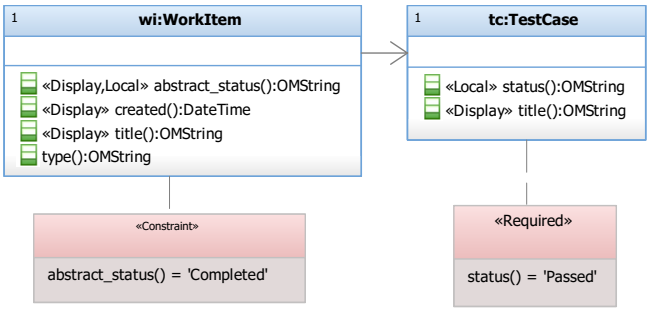


Fig. 3 Abstract visual notation for Rule **R5**

of the work item to the `abstract_status` local attribute, which contains the value `Completed`.

Other rules and queries shown above can be expressed using the same formalism. The stereotype «Forbidden» is the opposite of «Required», and signifies that the object or attribute is it associated with must not exist, and that constraints annotated with it must be false. The absence of either of these stereotypes marks simple query, which finds sets of elements that correspond to the query pattern. The «Smallest» stereotype is analogous to «Largest» but specifies the smallest value. Finally, associations between objects of the same type can be annotated with «OneOrMore», which means that the association can correspond to a chain containing several links of the same type.

5 Semantics

This section defines the semantics of visual queries and rules. A number of syntactic limitations are placed on the visual formalism; for example, a single query cannot contain both required and forbidden elements, except for

constraints. We define a *query* to be one that has neither required nor forbidden elements of any kind, and a *simple rule* to be one that has no required or forbidden objects, attributes, or associations, but may have required or forbidden constraints. A rule corresponds to a query that finds violations; a required constraint can therefore be translated into the negated form of the constraint, while for a forbidden constraint, a positive occurrence will be a violation. In this section, we will consider required and forbidden constraints as syntactic sugar for boolean conditions, and simple rules will be treated as queries.

The semantics of queries and rules are defined formally as a matching function that specifies when a set of RDF triples match the query. For the defininitions in this section, we assume that all type and property names have been replaced by the corresponding resources, as specified in the library. For queries (and simple rules), the matching function is defined as follows:

Definition 1 (Simple match). *A set of RDF triples from some repository* matches *a query or a simple rule if all the following conditions are met:*

1. *There is a one-to-one mapping m from objects and object attributes in the query to RDF resources or (in the case of attributes) values in the set. The notation $m(o)$ will denote the resource corresponding to object o, and $m(o, a)$ will denote the value corresponding to attribute a of object o.*
2. *The type of each object in the antecedent graph is the same as the RDF type of the corresponding resource (that is, for each object o of type T there is an RDF triple $\langle m(o), \texttt{rdf:type}, T\rangle$).*
3. *Let $o \xrightarrow{p} o'$ be an association in the query from object o to object o' labeled with property p. If the association does not have the «OneOrMore» stereotype, there is an RDF triple $\langle m(o), p, m(o')\rangle$ in the set. If the association is annotated with «OneOrMore», there is a sequence of resources $r_1 = m(o), r_2, \ldots, r_n = m(o')$ such that $n > 1$ and all RDF triples $\langle r_i, p, r_{i+1}\rangle$ are in the set.*
4. *If object o in the query has an attribute a of (primitive) type T, then the RDF triple $\langle m(o), a, m(o, a)\rangle$ is in the set, and $m(o, a)$ has type T.*
5. *If a query object o has an attribute a with the «Largest» (resp. «Smallest») stereotype, then there must not exist in the repository any resource r with a corresponding value of the attribute a (as in item 4) that is less (resp. more) than $m(o, a)$ such that for all associations $o \xrightarrow{p} o'$ or $o' \xrightarrow{p} o$, r is appropriately related to $m(o')$ (as in item 3), and all constraints associated with o have the same truth value for $m(o)$ and $m(o')$.*
6. *Each constraint in the query holds under the mapping m, except that constraints with the «Required» stereotype do not hold.*

For queries, this definition specifies all correct matches; these are the result of the query. For rules with forbidden elements, the same definition holds, except that any match corresponds to a violation of the rule. This is true since any match contains a forbidden object or a forbidden attribute.

For rules with required elements, it is necessary to distinguish between each rule's *antecedent* and *consequent*.

Definition 2 (Rule antecedent and consequent). *The consequent of a rule contains all required or forbidden objects, attributes, associations (but not constraints). The antecedent contains all other rule elements.*

The antecedent of a rule is required to be a connected subgraph; syntactically, it a query, and therefore simple match is defined for it according to Definition 1. Violations of the rule are those sets of triples matching the antecedent for which no extensions satisfy the consequent; in other words, there is no way to add triples from the repository so that all requirements are met.

Definition 3 (Rule match). *A set S of RDF triples from some repository* matches *a rule with required elements if all the following conditions are met:*

1. *The set matches the antecedent of the rule according to Definition 1.*
2. *There is no set S' of triples from the same repository such that $S' \supseteq S$ and S' matches the full rule according to Definition 1.*

6 Implementation

We implemented the visual formalism from Section 4 as a plugin to IBM Rhapsody [9], which enables editing of queries and rules in the visual formalism. The plugin uses the library to provide a palette of types, each with the relevant attributes. For each association, the plugin provides the list of applicable associations between the types of the two objects, based on the information from the library.

A query diagram is then translated into an equivalent textual description. We have also developed an Eclipse plugin to edit this textual formalism directly, with completion of types, attributes, and associations. This is shown in Figure 4 for Rule **R1**.

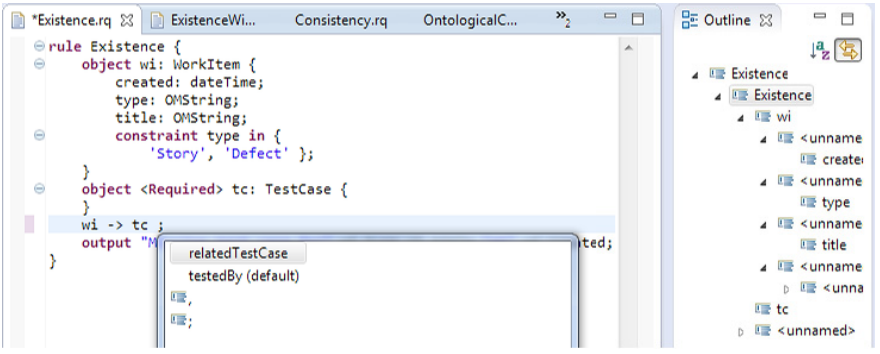


Fig. 4 Textual notation for Rule **R1** in the Eclipse plugin, showing completion for the abbreviated association name.

Another tool then takes the textual query and translates that into SPARQL, according to the semantics of Section 5. In experiments running one or more examples from each of the rule classes of Section 2 against two RELM repositories, violation were found for most rules. The library itself is extracted automatically from the repository, as explained in Section 3.

The extraction of the library from the two repositories takes a little over two minutes; compilation of a query to SPARQL is almost instantaneous; and running rules against both repositories takes a few seconds per rule, depending on its complexity.

7 Conclusion

This paper presented several classes of rules and queries over combined engineering data, and used these to derive a set of requirements from a query utility. In particular, engineers who are not software developers should be able to express their own queries, and abstractions over the RDF data must be provided for that purpose. A visual formalism based on UML/SysML object diagrams has been described. While similar in some respects to other visual rule notations [10], this formalism was tailored to be as close as possible to the familiar object diagrams, with rule-specific stereotypes.

The implementation contains a tool that extracts a library from existing repositories, uses the library to support convenient graphical editing in Rhapsody as well as textual editing in Eclipse, compiles queries into SPARQL and runs them against the repositories.

Acknowledgments. I am grateful to H. Broodney, M. Cantor, M. Cohen, Y. Dubinsky, A. Sela, and U. Shani for helpful discussions, and to S. Zolotnitsky for his help with the implementation.

References

1. OSLC: Open Services for Lifecycle Collaboration, `http://open-services.net`
2. Broodney, H., Shani, U., Sela, A.: Model integration — extracting value from MBSE. In: INCOSE Int'l Symp. (2013)
3. International Business Machines Corp.: Rational Engineering Lifecycle Manager, `http://www-03.ibm.com/software/products/en/ratiengilifemana`
4. Allemang, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann (2008)
5. World Wide Web Consortium: SPARQL 1.1 Query Language (2013), `http://www.w3.org/TR/sparql11-query`
6. International Business Machines Corp.: Jazz Foundation, `https://jazz.net/products/jazz-foundation`
7. Blochwitz, T., Otter, M., Åkesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., Olsson, H., Viel, A.: Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In: 9th Int'l Modelica Conference (2012)

8. Warmer, J.B., Kleppe, A.G.: The Object Constraint Language: Getting Your Models Ready for MDA. Addison-Wesley (2003)
9. International Business Machines Corp.: Rational Rhapsody family, http://www-03.ibm.com/software/products/en/ratirhapfami
10. Lukichev, S., Jarrar, M.: Graphical notations for rule modeling. In: Handbook of Research on Emerging Rule-Based Languages and Technologies. IGI Global (2009)

Leveraging Domain Expertise in Architectural Exploration*

Henry Broodney, Michael Masin, Evgeny Shindin, Uri Shani, Roy Kalawsky, Demetrios Joannou, Yingchun Tian, Antara Bhatt, and Imad Sanduka

Abstract. Domain experience is a key driver behind design quality, especially during the early design phases of a product or service. Currently, the only practical way to bring such experience into a project is to directly engage subject matter experts, which means there is the potential for a resource availability bottleneck because the experts are not available when required. Whilst many domain specific tools have attempted to capture expert knowledge in embedded analytics thus allowing less experienced engineers to perform complex tasks, this is certainly not the case for highly complex systems of systems where their architectures can go far beyond what a single human being can comprehend. This paper proposes a new approach to leveraging design expertise in a manner that facilitates architectural exploration and architecture optimization by using pre-defined architecture patterns. In addition, we propose a means to streamline such a process by delineating the knowledge creation process and architectural exploration analytics with the means to facilitate information flow from the former to the latter through a carefuly designed integration framework.

Henry Broodney · Michael Masin · Evgeny Shindin · Uri Shani
IBM Research, Haifa, Israel
e-mail: henryb@il.ibm.com

Roy Kalawsky · Demetrios Joannou · Yingchun Tian · Antara Bhatt
Southborough University, UK
e-mail: r.s.kalawsky@lboro.ac.uk

Imad Sanduka
Airbus Group Innovations
e-mail: imad.sanduka@airbus.com

* This work is partially funded by the FP7 DANSE project (grant number 287716), and the SPRINT project (grant number 257909).

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_7

1 Introduction

The design of complex multi-disciplinary systems, especially Systems of Systems (SoS) [11], begins with a set of requirements that lead to the definition of high-level architecture. In this process the overall topology is created, the main building blocks and the relationships between them are identified, and the main design metrics (e.g. cost, performance, complexity) are established. Unlike many detailed design tools, where the embedded analytics (e.g. synthesis, simulation and optimization) help engineers to leverage the experience of many experts that have contributed to the creation of these tools, the systems engineers tend to rely on personal and immediate peers' knowledge to accomplish their tasks.

This aspect creates several problems, including the preservation of knowledge within an organization which is frequently tightly coupled to the continued employment of the specific individuals. Even shifting such individuals between domains within the same company can be problematic and leads to corporate memory loss. Consequently, project schedules and quality are greatly affected by the need for new engineers to re-learn what their predecessors already knew. Moreover, even for the experienced engineers there is still a requirement to learn new things, as technologies or methods evolve, or the engineers shift to new roles involving systems with different nature from what they have previously encountered.

This challenge is inherently multi-faceted and multi-disciplinary in nature. One important aspect is to provide the means and methodology for capturing required expertise. The most straightforward approach would appear to rely on producing documents outlining best practices, examples of their usage, do's and don'ts, and so on. Indeed such approaches are often used in industry, but this has its shortcomings because such document driven processes allow for much human interpretation and assumes lots of cross-discipline understanding from the reader. Also, the quality of the document text has great influence in terms of the original purpose and context it was produced for. The evolving Model Based Systems Engineering (MBSE) methodologies are attempting to address these shortcomings by providing formal mathematical means to capture and use knowledge [28]. This paper describes our research to extend MBSE through the use of architectural patterns in order to capture and reuse best practices for building large SoS. In addition, architecture patterns can include underlying mathematical formalisms to facilitate computation of the main design metrics for the instances of the patterns. These approaches have significant potential to capture the experts' knowledge and facilitate effective deeper understanding of the system, its context, purpose, implementation and limitations [25]. We propose an integration framework capable of applying knowledge in one set of formalisms and furnishing this knowledge in another set of formalisms, whilst making sure the semantics of the originating knowledge is uncompromised. Furthermore, this approach can lessen the problem of searching for the required knowledge, because it removes the dependency between the source and the destination of the information flow to be in the same

domain. The approach is illustrated in this paper by means of a simplified exemplary Use case from the authors' research in the context of the DANSE EU FP7 project [11].

2 Engineering Complex Systems with Architecture Patterns

The use of patterns in certain areas of engineering (such as software engineering) is well established [16][17]. A foundational work on patterns [27] refers to recurring structures, objects and events that can be used as designs, blueprints, models or templates in the creation or representation of the architecture of a complex system. In the latter case, newly created entities inherit the characteristics of the parent object (pattern). Architecture patterns [26] are an extension to the patterns concept and can be used as the starting point to lay basic foundations of a system's architecture, and which can evolve or be refined during its lifecycle. Architecture patterns can incorporate practices that have proven successful in the past. It is important to note they are not prescriptive, but suggestive by including guidance on when their use is most appropriate and provide examples from existing proven systems. Consequently a pattern has structural and dynamic properties whose form is realized through a finite number of visible and identifiable components. A component in this context can be technical or non-technical entities, services or even software. The notion that a pattern [18] is a general repeatable solution to commonly occurring problems makes pattern reuse a big step in reducing system design risks. When used correctly, patterns provide an explicit way to articulate common concepts at the operational through to the detailed implementation levels. Additionally, their use eases the burden of characterizing a complex system for analysis studies but the approach should not be under-estimated. The growing importance of patterns has led a number of new initiatives such as the INCOSE/OMG Model-Based Systems Engineering (MBSE) Initiative leading to a Pattern-Based Systems Engineering (PBSE) Challenge Team establishing a draft charter for the advancement of patterns (http://www.omgwiki.org/MBSE/doku.php). The use of patterns relies on the mining of architecture patterns from existing (or legacy) systems in conjunction with access to a comprehensive patterns repository (searchable library). Details of how to mine architecture patterns are provided in [26]. However, it is important to note that architectural patterns are conceived at the higher operational level whereas design patterns (familiar to the software engineering community) are applicable at the system and lower levels. An understanding of patterns provides important benefits, particularly in that they provide a common language, which is independent of the underlying technology. This is highly dependent on the quality of the pattern being used and how the pattern is to be deployed in subsequent system modeling and analysis through simulation.

There are four key processes involved in the use of patterns for architecture design, i) the creation of patterns, ii) pattern selection, iii) refinement of the pattern for subsequent use/deployment within the architecture under consideration and iv)

selection of the optimal architecture based on a multi-criteria decision analysis technique. It should be noted that architecture design is a highly creative process that symbolizes the 'function' of a product, system or SoS into a form through a concept. In the first instance, experienced systems practitioners attempt to extract specific patterns since they have the domain knowledge of what is important (and potentially re-usable) – the danger the less experienced person may fall into is expressing patterns in too much detail such that it becomes too implementation specific rather than more generally usable. The expert systems architect applies their understanding (in order to deal with aspects such as uncertainty, ambiguities and contradictions) on how to abstract a complex system into its constituent parts at an appropriate level. While the first step is inherently "expert-based" the other steps, i.e., pattern selection, refinement and optimization, the goal is to facilitate this automatically using concepts covered in this paper.

As architecture patterns are created they are stored in a searchable online repository that is accessible via the systems architecting modeling tools. Architecture patterns may be created from pre-existing patterns, mined from the original ensemble of legacy systems comprising the complex system or mined from other domains but which are relevant in the current application. Where the system architecture is required to evolve over time the initial set of patterns can usefully represent the starting state of the system's architecture and enable performance analysis to be undertaken during future modification of the system. A key step in developing architecture patterns is to include performance data with the pattern information since this will be required in order to evaluate future generations of the architecture of the complex system. This makes it easier to see how the complex system has evolved over a longer period of time. When implemented correctly, the performance metrics associated with each pattern helps with automating the process of finding Pareto optimal system architecture configurations.

The overall flow for the proposed process and method is outlined in Fig 1. The architecture patterns and associated components' data are published into the integration framework and stored within it. The optimization tool extracts the patterns from the integration framework in the language and format the tool requires, along with additional data needed for the optimization process. The resulting system models are then published back into the framework for potential usage by other stakeholders (outside the scope of this paper) to support other architecture analysis needs.

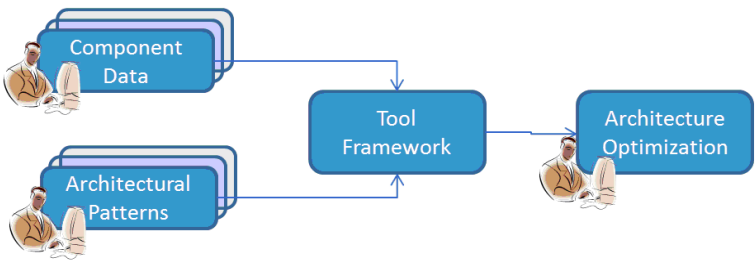


Fig. 1 Proposed tool flow

It is important to note that architecture patterns should, wherever possible, be independent of the specific implementations since this would render them less transferrable. Care should be taken to avoid creating huge (overly complex) patterns since these are less usable, instead patterns should be structured into smaller more recognizable elements. At the level of architecture patterns it is better to assume the set of patterns is recursive in the sense that they form a hierarchical structure comprising patterns and lower-level patterns. Patterns for different layers are usually defined in different tools and languages, such as the Architecture Analysis and Design Language (AADL), Systems Modeling Language (SysML), or Unified Modeling Language (UML). All these patterns should be converted to the analysis/optimization tool semantics for potential architecture topologies as discussed in Sections 3 and 4.

In order to explain the process behind architecture patterns (refer to Fig 2), the mining process and application is illustrated by means of a simplified example within the (system of systems) context of a communication system antenna upgrade for an emergency response system.

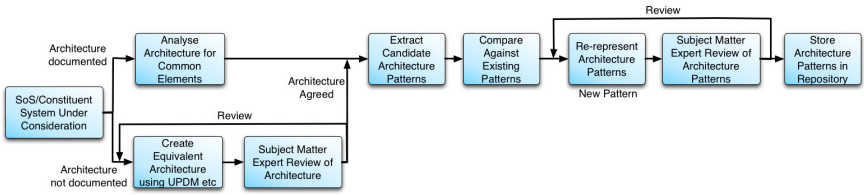


Fig. 2 Outline of the architecture pattern mining process

3 Architecture Patterns – Example Use Case

To illustrate how architecture patterns can be used we consider a simple example involving a communication system for an emergency response (ER) scenario, its services and other critical parts of the system responsible for interactions between the constituents of a system of systems (SoS) (unfortunately, CSDM paper page limits make it impractical to consider the wider ER SoS). The requirements of the communication system directly affect the architecture of the system. In addition, it is important to consider the evolution of the communication system and its effect on emergency response operations and the synthesis of its constituent systems. In this example, the emergency response system of systems consists of hardware, software, and human systems distributed over multiple operational nodes i.e. command and control centers, fire brigades, police headquarters and hospitals. These systems interact and exchange information between each other in order to deal with emergency cases, deploying resources such as people and emergency appliances to deal with the incident. The operational nodes tend to be autonomous, geographically distributed and evolutionary developed systems with communication capabilities that effectively comprise a typical SoS structure. Although these

systems belong to the overall SoS together with services systems (e.g. communication systems), they still retain their own goals, objectives and constraints; meanwhile the overall SoS must achieve its own goals at an optimal operational level. The purpose of the Use case example in this paper is to help illustrate the process of synthesizing possible SoS architectures that represent the communication technology evolution within the SoS and provide an optimized solution for technology transformation. It is not the intention of this work to replace the need for the highly detailed tools that are required later in the deployment phase to calculate actual antenna performance from effects such as terrain screening etc.

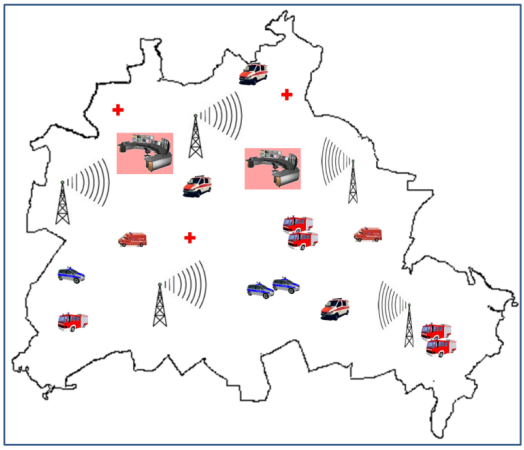


Fig. 3 Antenna Placement Use Case

In the example (refer to Fig. 3), it has been assumed that as a starting point, emergency response systems use Terrestrial Trunked Radio (Tetra) as the current technology for dedicated emergency communication systems. A future consideration is the transition to 'Long term evolution' (LTE) to provide greater capability for date exchange, with the potential to enhance overall SoS operations and services. Reference to Fig. 4a and 4b show the specific architectural differences between the Tetra and LTE systems.

The services components of each system are very different but for the purposes of this example the requirement to place antenna systems in optimal locations is the same. The key differences to note is that the operating range, frequency, effect of terrain and cost is very different between the two systems. The purpose of this Use-case is to show how an optimized solution for the transition from Tetra to LTE technology can be achieved by considering changes that must be implemented on the constituent systems comprising the SoS and ensuring maximizing the overall benefits of the new technology. Since this is a large task this paper is restricted to considering the optimization and best placement of new antennas or replacement for old antennas.

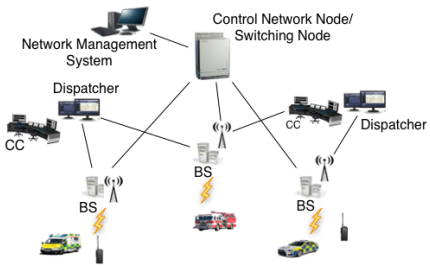


Fig. 4a Top-level architecture patterns for Tetra communication systems

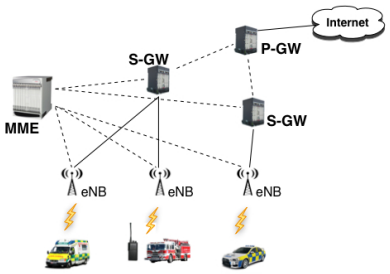


Fig. 4b Top-level architecture patterns for LTE communication systems

The architecture synthesis process must consider different architecture solutions and technical constraints such as:

- Hardware constraints of the current systems and their adaptability
- Geographical distribution of the constituent systems (including effects of terrain, rural and urban requirements)
- Possible antenna position limitations (location in sensitive areas, security, interference)
- Technical properties of new system (coverage (range), power, quality of service, etc.)
- Redundancy (ensuring reliable communications in the event of failures etc.)

In addition to the architecture constraints, the following optimization constraints and objectives must be considered:

- Lowering communication cost
- Achieving best communication coverage over the specified area
- Provision of the best communication quality and services for the whole SoS

At a deeper technical level, it is important to consider alternative choices for antenna types, operating frequencies/channels, height, and positions for both Tetra and LTE systems in conjunction with the required hardware components that support the communication systems. Consequently, of interest is the optimized SoS architecture solution that describes:

- Number of required antennas and their properties
- Position of the antennas (placement of new antennas or replacement of old ones)
- Constituent systems and hardware components of the new architecture
- SoS Architecture description

The current optimization solutions for the use case aim to provide an optimized solution for antenna coverage, placement, and cost as a separated system without

considering the interaction and the effect on the other systems within a SoS. The proposed optimization process integrates properties of all constituent systems and connections within the SoS architecture along with relevant SoS metrics. Also, it automates the SoS design and development process and reduces the cost and development effort when considering the overall effect and interaction of the systems together within one architecture model, rather than considering the evolution and development of each system separately which has resulted in interoperability problems in the past. In the case of SoS optimization the optimization objectives are distributed between global and local goals and considered together in the development of the optimized architecture model.

3.1 Antenna Architecture Patterns

It is worthwhile noting the different architectural solutions that must be considered when designing the antenna system for an emergency response SoS. At its basic level, the antenna system corresponds to the well-known pattern for a cellular network where each mobile communication device (personnel equipment or vehicle based) communicates with the antenna system closest to the user. When the user goes outside the coverage of this antenna the system switches over to the next nearest antenna system (or cell).

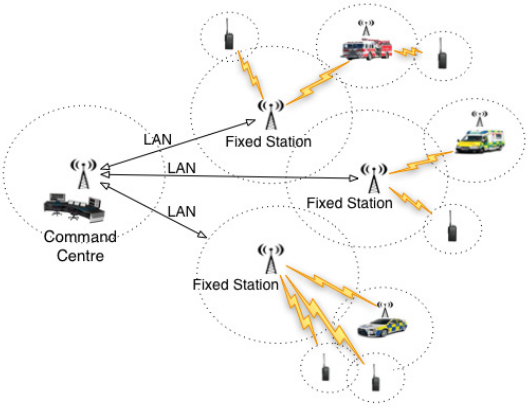


Fig. 5 Cellular Communication Network Antenna Pattern

There are many variants for the above architecture. For example, consider the case where one of the cellular nodes is inoperable or where communication coverage is particularly bad. In this case, it is feasible to consider setting up a temporary mobile communications node. This could be achieved by setting up a separate portable communications node (relay) or using the higher-powered transceivers on say a fire engine as the relay station. Whilst overall coverage may be restricted due to the lower antenna height of the vehicle it could still provide perfectly

usable communications whilst dealing with an emergency incident. Note, in this situation the connection between the command center and the relay is unlikely to be achieved through the local area network (LAN) and instead rely on a separate radio frequency (RF) channel.

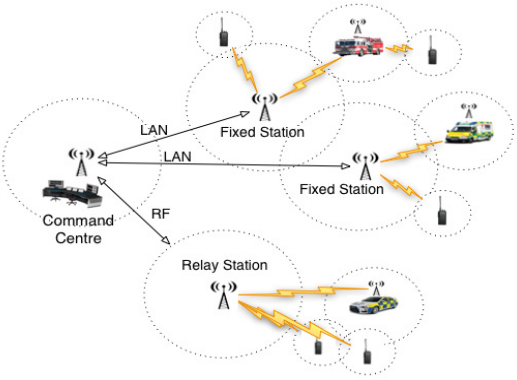


Fig. 6 Cellular Communication Network Antenna Pattern - Relay

By drawing upon the different architectural configurations (patterns) and including their different performance metrics into the optimization process it is possible through a technique of multi-criteria decision analysis to trade performance against cost and other factors to determine optimal antenna placement.

4 Integration Framework

In this section the problem of creating an integrated flow of modeling data among tools through which the variability in the initial model is conceptualized and limited to specific optimal values through the optimization stage is discussed. This flow requires an operational multi-tools environment whose provision represents a significant problem in model based system engineering – the large plethora of design tools that are required to support a complex engineering development project. An integrated tool environment is required where different tools interoperate enabling information created in one tool to flow to other related tools. This is known as the tools interoperability problem, and is further complicated since there are many different functional areas in systems engineering, each of which has its own way of specifying what is needed, and in what representational language.

Over recent years a number of standards have evolved to bridge the gaps between tools. For example, the Open Service for Lifecycle Collaboration (OSLC) [4] made progress with very promising attempts to cover all tools in all aspects of the development process, or the product life-cycle. In our example optimization scenario, we require patterns, data and an architecture design in which multiple/alternative options can be constructed and evaluated by an optimization

system. In this respect the importance of interoperability via OSLC should be obvious. The problem which OSLC solves is mostly in defining the services that a tool needs to provide in order that other tools can query it and obtain the modeling data that is manages. Unfortunately, OSLC currently lacks the detailed semantic information to allow proper transformation of modeling data from one tool to the language and semantics of another tool. However, OSLC adopting the semantic web technologies has been an important initial step in this direction. The semantic web technologies include the use of Resource Description Framework (RDF) [5] for model representation, RESTful protocols [6] for communications, and the concept of Linked Data [7] to link between modeling elements (i.e., "Resources") of the different tools. Once a model is represented in RDF, it is conveniently 'query-able' with the SPARQL [8] query language.

The semantics of models is formally defined using ontology described in the Web Ontology Language (OWL) language standard [9], a description that is also represented in RDF to provide an actionable and query-able specification of the models in the different tools. The ontology of a tool is its "Modeling Language". When a tool can export a component from its internal repository to RDF according to some formal ontology – that model can then be processed by other tools. Since each tool has its own language, and hence its own ontology, we can flow models having different ontologies between tools using a transformation engine that mediates the models. This leads to a network of mediators – see Fig. 7, which connects repositories, each of which is governed by an ontology. Tools contain model

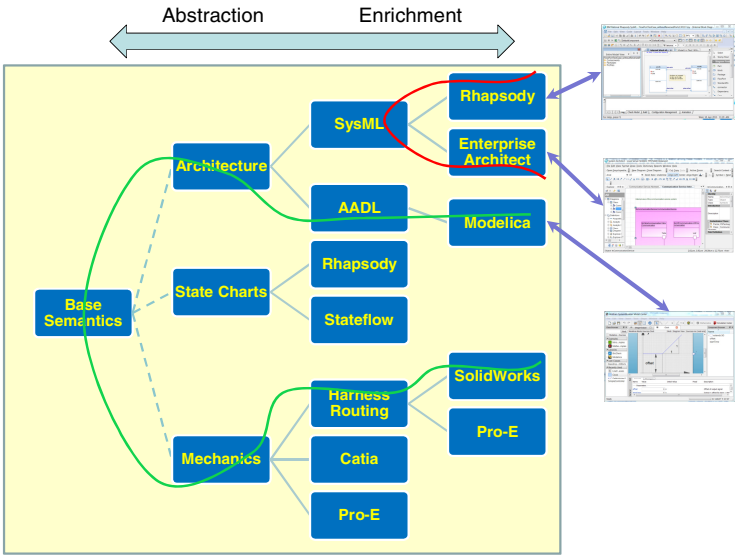


Fig. 7 Mediation network with a hierarchy of modeling ontologies

data in their own proprietary format and representation, and each tool exposes that into the ecosystem at a standard format which is common and easily usable by any compliant tool. The mediation network we call "Semantic Mediation" bridges the semantic gaps among tools.

In Fig 7, each block represents a repository in which models of a certain representational language are stored. The language of a given repository is shown by the name appearing on that box. It can be seen that a "Rhapsody" repository and an "Enterprise Architect" repository are each connected with their respective tool. The representation of content in these tools is proprietary, but once shared with the integrated tool environment platform, their content are stored in RDF repositories, according to a specific ontology. That RDF is already an "open" representation of the model that can be shared.

These two repositories are connected via mediators to a third block labeled "SysML". SysML [12] is a standard specification which is used by both IBM Rational® Rhapsody® [13] and Enterprise Architect™, enabling it to serve as a "common" language for both tools. Mediation from a specific tool's ontology with a common language is not difficult. Unfortunately, not all properties of each tool will necessarily come through the transformation process. (After all, the specific features of each tool are what makes if unique and valuable to the engineer.) This is the fundamental concept of the semantic mediation – common features represent less than the whole of each of the relevant languages and tools. This concept is more dramatically demonstrated in the green path (N.B. for readers using monochrome printouts of this paper this is shown as a bold line) in Fig 7 where four levels of mediation are exercised, going down to what we term "Base Semantics" on the far left. At this level, the only common feature are the resources (i.e., design elements) themselves with some basic properties such as name and description. It is recognized that even this is valuable information to share. In fact, OSLC domains are at a low level of minimal commonality, and the "Architecture Modelling" domain of OSLC (in short OSLC-AM) is basically at that basic semantic level.

The semantic mediation technology is described in further detail in [2] and [3], and has been developed in the SPRINT project [10] and has also been exploited in the DANSE project [11]. As demonstrated in Fig. 7 each tool has only one interface to the mediation environment, which makes system extensibility very easy, limiting the impact of a change anywhere in this network, allowing adding the N^{th} tool without affecting other tools already integrated. That is a strong concept which complies with what is known as "Hub and Spoke" architecture – using a hub of services of the semantic mediation platform. This interoperability platform is implemented on top of the IBM Jazz® tools suite [14], but is still a research tool rather than an off the shelf product.

In the next subsections a description is given on how use case integration is enabled through this mediation platform, taking model data from Rhapsody and Microsoft Excel to accomplish an optimization processes.

4.1 Architecture Pattern Representation

Architecture patterns are typically expressed as models, whose ontology accommodates variability. A special modeling tool is typically used, or one can extend an existing tool. In the example selected for this paper SysML [12] is illustrated, which was extended with a special profile for architecture patterns. Therefore, the process starts with an experienced engineer building up architecture patterns in the methodology as described in [1]. Once the patterns are created, they can be stored in the interoperability platform as an RDF graph according to the architecture pattern ontology. Any number of appropriate patterns and their models within the repository can be imported into a design tool for building up the concise model. Sharing the model between these profiles does not require any special development on Rhapsody, except for the sharing capabilities with the single interoperability platform, which treats these two independent capabilities in Rhapsody as distinct tools having their own languages. Other tools can also be used in this process as long as we can mediate models from the other tools with the architecture pattern ontology. The systems architect selects and applies the patterns within the Rhapsody profile. The next step is to process the concise models [26]. To support the optimization process, additional metrics data is required which is related to the patterns selected to be evaluated, this is described in the following section.

4.2 Data Supporting Architecture Optimization Process

Data supporting the architecture optimization process relates to the measurable quantitoies (typically expressed numerically) to be applied to the variability of the patterns. The most convenient tool to use for data entry in this respect is Microsoft Excel (Excel has the advantage of being one of the most popular tools for engineers in all domains of engineering). Excel being a tabular environment, facilitates simple mapping from tables and models as follows:

- Each table correspond to a class in an ontology
- Each row in a table is a “thing” or instance of that class
- The columns are the properties of instances, and the set of properties in a table define the properties of a class in the ontology

In the example discussed in this paper the properties are all ‘Data’, rather than references to other instances of any of the classes. Each table being created on a separate Excel spreadsheet allows a simple tabular schema to be generated automatically and completed by the engineer to enrich the concise model with data.

Both of these aspects of the data management can be automated through the interoperability platform and its concepts. We progress as follows:

- Provide ontology for the schema of an Excel file
- Mediate architecture pattern to a model in that ontology
- Import the ontology to an Excel definition agent which will create an Excel file in which this schema is defined

- The systems engineer user completes the spreadsheet and exports it to the interoperability platform as a model of an ontology, which is derived from the Excel schema being mediated
- The concise modeling tool can import both pattern

The data populating the spreadsheet can also originate from other sources or tools, but it should be related to the modeling terminology derived from the architecture pattern description.

5 Architecture Optimization

One of the main activities of the Systems Engineer is the creation of alternative SoS architectural solutions with recommendations on optimal solutions. These architectures must satisfy all SoS and 'constituent systems' requirements and goals. However, the ever-increasing complexity of systems, strict design constraints, conflicting goals, and many other factors means the process of finding optimal design is an extremely difficult task. The common means to achieve SoS goals is to build optimization models for the set of specified architectures and find the best one using suitable optimization software tools. Unfortunately, this approach is highly labor and expertize intensive, because each architecture solution created by the engineer requires a separate optimization model created by an expert. The issue can be successfully resolved and demonstrated by using the approach described in [1] and [15]. The systems engineer can rapidly create the necessary system architecture satisfying all functional and technical constraints and achieving specified goals. However, even in this case, optimization can be accomplished only in the scope of a selected architectural pattern through its own metrics and constraints. These metrics and constraints are derived from the respective system domains utilizing domain-specific knowledge, which can be unfamiliar to the engineer. However, within the proposed framework we provide a mechanism to capture this domain specific knowledge and bring it to the SoS models.

The tool net mechanism described in Section 3 provides the engineer with a set of functional patterns together with set of technical patterns and set of mappings between them, where each functional pattern mapped to one or more technical patterns. Each triad (functional pattern, mapping, and technical pattern) together with corresponding data catalogs and (in this use case) geometry can be treated as separate concise model [1]. The information captured by the architectural pattern is represented by various types of concise modeling elements. The concise model, further enriched by additional elements, goals and constraints provided by the engineer, is automatically translated to corresponding optimization model. At the next step, a set of optimization problems is solved and the set of optimal solutions is ranked and filtered according to selected goals. Finally the systems engineer receives a set of Pareto-optimal architectures and can subsequently select the most appropriate solution.

The process can be illustrated on the communications system use case described in section 2. In this case we utilize following domain specific knowledge:

- Geographical knowledge utilized by existing communication system antennas disposition, possible places for new antennas and maximum numbers of antennas in selected positions
- Radio-electronic knowledge utilized by coverage tables, communication equipment types and possible equipment connections

In the simple use case only one architectural pattern was described, but was sufficient to demonstrate the related paradigm.

Captured architectural pattern translated into SysML model with concise profile extension [1]. Captured domain specific data translated into corresponding Excel file and geometrical layer elements.

For the use case, Fig. 8 shows the main parts of the potential communication system topology in SysML internal block diagram.

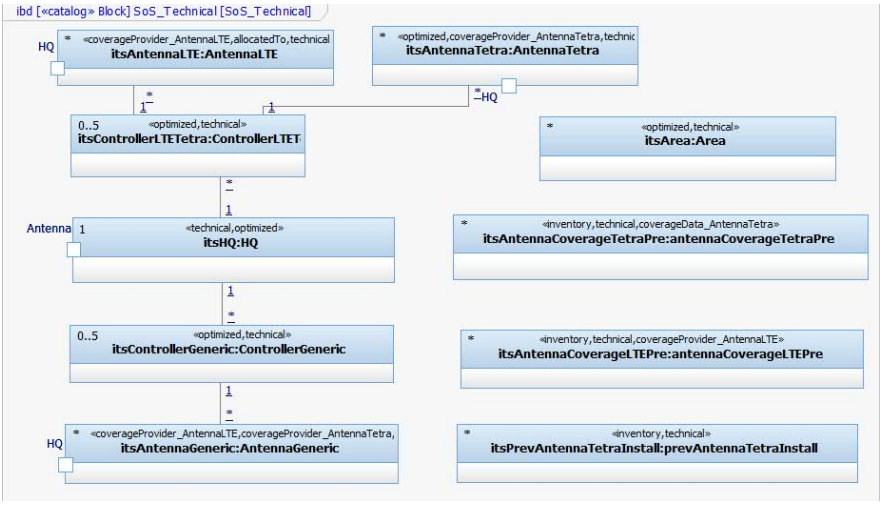


Fig. 8 Communication system technical (Internal Block Diagram) IBD

The diagram utilizes following domain specific knowledge:

- There are coverage area to be covered by two types of mobile networks (Area)
- There are 3 different types of antennas: one can be used in LTE network only (Antenna LTE), one can be used in Tetra network only (Antenna Tetra) and one can be used in both networks simultaneously (Antenna Generic).
- There are 2 different types of controllers: one to control LTE and Tetra antennas only and one to control Generic antennas only
- Each antenna must be connected to one controller, which is placed in the command center

- Number of antennas connected to one controller dependent on controller model
- The coverage data for both types of networks, provided by corresponding coverage tables

The systems engineer adds to the technical Internal Block Diagram (IBD) the additional component representing the existing Tetra network infrastructure. The engineer also provides the necessary constraints on the technical block attributes related to the coverage of both networks and the cost of the projected system, defines the optimization goals and starts the optimization process. After the process ends the systems engineer obtains optimal architecture represented by back-annotated SysML diagram represented in Fig. 9. The back-annotated SysML diagram represents the optimized version of the SoS architecture. The term 'back-annotated' refers to the concept that the architecture has been derived from the original architecture and can be re-used as a future architecture solution.

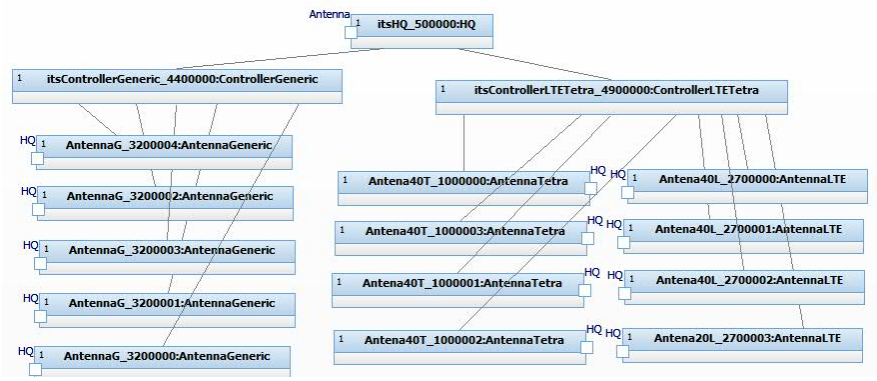


Fig. 9 Communication system optimal architecture

6 Summary and Future Directions

In more traditional engineering approaches the ability to optimize a given architecture has been extremely difficult. Moving away from traditional document centric solutions to the model based approaches described in this paper will facilitate a more complete representation of complex systems that would be otherwise impossible to address by document based approaches. This paper has illustrated how architecture patterns representing alternative architecture solutions can be incorporated into an analysis process to facilitate multi-criteria design space optimization for SoS. This process is generally regarded as very challenging for complex systems (SoS) but the approach described in this paper is beginning to show great promise for tackling large-scale systems. Our experience to date is highlighting the diversity and wide application for this approach. Although we have

worked with a set of mature tools (such as IBM Rhapsody) where much of the required data integration has already been integrated, it is perfectly feasible to apply the approach to other tool sets through the semantic mediation facility. In our current research we are looking into applying reuse of patterns as well as components of design using this capability. We are also looking at capturing sociological and ethical aspects of the patterns and in particular how to capture such expertise. The intention is to ensure that experts' knowledge is considered earlier in the design process. This is particularly important for SoS and their legacy constituent systems that have a long service life where initial expert input is no longer available.

Acknowledgments. This work was supported in part by European Commission for funding the Large-scale integrating project (IP) proposal under the ICT Call 7 (FP7-ICT-2011-7) 'Designing for Adaptability and evolutioN in System of systems Engineering (DANSE)', contract 287716, and the Specific Targeted Research Project (STReP) SPRINT – 'Software Platform foR INtegration of engineering and Things', contract 257909.

References

1. Broodney, H., Dotan, D., Greenberg, L., Masin, M.: Generic Approach to Architecture Optimization in MBSE. In: INCOSE International Symposium 2012, Rome, Italy (2012)
2. Broodney, H., Shani, U., Sela, A.: Model Integration – Extracting Value from MBSE. In: MBSE, INCOSE International Symposium, Philadelphia, U.S (2013)
3. Shani, U., Wadler, D., Wagner, M.: Engineering Model Mediation which Really Works. In: INCOSE'IL 7th International Symposium 2013, Herzliya, Israel, March 4-5 (2013)
4. (OSLC) – Open Services for Lifecycle Collaboration, http://open-services.net/
5. (RDF) Resource Description Framework RDF, http://www.w3.org/RDF/
6. (REST) Representational State Transfer, http://en.wikipedia.org/wiki/Representational_State_Transfer
7. (W3C) "Linked Data", W3C Standard, http://www.w3.org/standards/semanticweb/data
8. (SPARQL) SPARQL Protocol and RDF Query Language, http://www.w3.org/TR/rdf-sparql-protocol/
9. (OWL) W3C OWL Web Ontology Language, http://www.w3.org/TR/2008/WD-owl2-new-features-20081202/
10. (SPRINT 2010-2013) Software Platform for Integration of Engineering and Things, http://www.sprint-iot.eu/
11. (DANSE 2011-2014) Designing for Adaptability and evolutioN in System of systems Engineering, https://www.danse-ip.eu/home/
12. (SysML) SysML specification version 1.2, http://www.omg.org/spec/SysML/1.2/
13. (Rhapsody) Rational® Rhapsody®, http://www-142.ibm.com/software/products/us/en/ratirhapfami/

14. Jazz, http://jazz.net
15. Masin, et al.: Pluggable Analysis Algebras for Design Space Exploration. In: CSER (2013)
16. Buschmann, et al.: Pattern-Oriented Software Architecture: A System of Patterns. Wiley (1996)
17. Gamma, E.R.J., Helm, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Orientated Software. Addison-Wesley, Massachusetts (2008)
18. Riehle, D., Züllighoven, H.: Understanding and using patterns in software development. Theory and Practice of Object Systems - Special Issue on Patterns 2, 3–13 (1996)
19. Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., Halkidis, S.: Design Pattern Detection Using Similarity Scoring. IEEE Transaction on Software Engineering 32 (November 2006)
20. Wendehals, L.: Improving design pattern instance recognition by dynamic analysis. In: Proceedings of the ICSE Workshop on Dynamic Analysis, pp. 29–32 (2003)
21. Zhang, Z., Li, Q., Ben, K.: A new method for design pattern mining. In: Proceedings of the 3rd International Conference on Machine Learning and Cybernetics (2004)
22. Huang, H., Zhang, S., Cao, J., Duan, Y.: A practical pattern recovery approach based on both structural and behavioral analysis. Journal of Systems and Software 75, 69–87 (2005)
23. Dong, J., Zhao, Y., Peng, T.: Architecture and Design Pattern Discovery Techniques – A Review. In: Proceedings of the 6th International Workshop on System/Software Architectures (IWSSA), USA, (2007)
24. Dong, J., Sun, Y., Zhao, Y.: Design Pattern Detection By Template Matching. In: Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC), Ceará, Brazil, pp. 765–769 (2008)
25. Weyns, D.: Architecture-Based Design of Multi-Agent Systems, ch. 3, pp. 27–53. Springer, Heidelberg, doi:10.1007/978-3-642-01064-4_3
26. Kalawsky, R.S., Tian, Y., Joannou, D., Sanduka, I., Masin, M.: Incorporating Architecture Patterns in a SoS Optimization Framework. In: Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013, Manchester, pp. 1726–1731 (2013)
27. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977)
28. Pfister, F., Chapurlat, V., Huchard, M., Nebut, C., Wippler, J.-L.: A proposed metamodel for formalizing systems engineering knowledge, based on functional architectural patterns. Systems Engineering 15, 321–332 (2012)

Seven Issues on Distributed Situation Awareness Measurement in Complex Socio-technical Systems

Maria Mikela Chatzimichailidou, Angelos Protopapas,
and Ioannis M. Dokas

Abstract. Distributed Situation Awareness is the modern perception of Situation Awareness that considers it as an emergent property of a complex socio-technical system rather than an individual endeavor. Although there are a plethora of Situation Awareness models and measurement techniques, it is doubtful whether they mirror the critical elements and behaviours of real collaborative and complex environments. This paper collects some of the most crucial issues surrounding the existing SA measurements techniques that arose under the complex socio-technical systems settings, and along these lines it addresses the need to change the paradigm in order to prepare the ground for a future Distributed Situation Awareness assessment technique, moving beyond the existing network-based approaches.

1 Introduction

In order to achieve their purposes, complex socio-technical systems involve various interactions between humans, machines, and other actors possibly coming from the outer environment of the system. Within the predefined boundaries of socio-technical systems, all elements and agents affect, and are affected by, the system's overall behaviour, thus they need to be looked at as an entity. There is therefore a shift from an analysis based on system decomposition, to an analysis that looks the system as a whole, when the objective is to examine the behaviour of a complex socio-technical system in terms of different inner or outer phenomena and/or stimuli. In such systems, Situation Awareness (SA) arises as a crucial requirement to achieve their higher goals.

Maria Mikela Chatzimichailidou · Angelos Protopapas · Ioannis M. Dokas
Democritus University of Thrace, Department of Civil Engineering,
Vassilissis Sofias 12, 67100 Xanthi, Greece
e-mail: {mikechat,aproto,idokas}@civil.duth.gr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_8

Endsley [1] defines SA as a state of knowledge of an individual; SA depicts how much, and how accurately, humans are aware of their current and/or systems situation and it concerns (a) the perception of the elements within a system, (b) the comprehension of their meaning, and (c) the projection of their future state.

In socio-technical systems, the complex links of responses and feedback between agents, elements, hierarchical levels, and (sub)systems, which all exhibit dynamic behaviour, are important for the formation and transformation of SA, since SA is not in a steady state. Hence, the Distributed SA (DSA) approach accounts for SA in collaborative environments [2, 3] and holds that the socio-technical system is the unit of analysis [4], because it treats SA as an emergent property of the complex socio-technical system.

In Figure 1, Salmon *et al.* [2] give a DSA example, where DSA is denoted by the big 'bubble' and integrates subsystems and a complex network of their in-between links. Within the big circle, each cluster is bounded by a smaller dashed circle, i.e. open boundaries that serve communication purposes, and represents individual SA affected by other's individual SA. Within these clusters, there are human-like figures representing human agents, there are document- and computer-like objects representing elements that convey information and data to the elements with which, and/or whom, they are affiliated. The technology-to-technology, human-to-human, and artifact-to-human links represent the communication channels that aid the efficient data and information acquisition to the required human and/or nonhuman system elements in real-time setting.

Naderpour *et al.* [5] state that any given system should be able to support the SA of its agents, i.e. the group of system elements that possesses

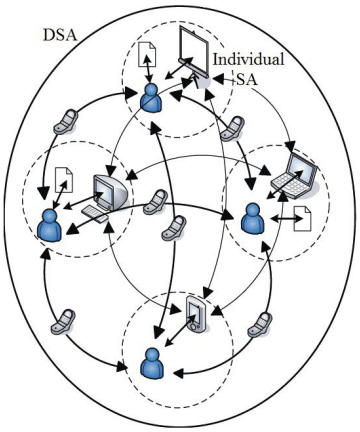


Fig. 1 Distributed Situation Awareness example [2]

reasoning mechanisms and demonstrates a capability to influence other agents, elements, and/or situations. Respectively, the ownership of information is initially at a system level [2], but it can be used, passed, and shared by agents to shape their own picture about the current situation of the system and, through this, support the emergence of DSA required for the system to function effectively. Furthermore, each information package held by one system element, e.g. agents, computers, documents, displays etc., serves to modify, and is modified by, other element's information. Therefore, DSA is held by the entire system since no one system agent has a complete picture of any situation, but just a facet of the situation at any point in time [2].

The overabundance of SA models, e.g. individual, team SA etc, is the evidence that SA is one of the most discussed and complicated cognitive processes in the field of socio-technical systems. Accordingly, the need to measure SA in man-made systems has led scientists to introduce a number of SA measurement approaches. Despite the intense activity regarding SA-related matters, Salmon *et al.* [3], Stanton and Young [6], and Salmon *et al.* [7] claim that there is criticism associated with the existing SA measurement techniques and their accordance with the disciplines, i.e. hierarchy, emergence, communication etc [8], of complex socio-technical systems. In particular, they argue that there is little evidence that these measurement techniques actually work [7], and they also raise concerns about their reliability and validity in cases where the objective is to measure SA in complex socio-technical systems. Furthermore, Salmon *et al.* [3] and Salas *et al.* [9] have stated that the existing (and exhausted) individual and team SA models and measurement techniques are proven not to be adequate in the context of modern complex systems. The DSA concept, however, sets the foundations for a systemic framework for explaining the emergence of SA, contrary to individual and team SA models, which only partially explain the SA formation, missing the notion of the emergence of SA as a system property.

This paper argues that the existing SA measurement techniques are not capable of measuring DSA and identifies some of the most challenging issues that render the existing SA measurement techniques inadequate for the purpose of measuring DSA in a complex, multi-hierarchical, and multi-agent environment. Thus, in the Conclusion, this paper provides a vision for researchers to assess DSA in the context of complex socio-technical systems, as a possible solution to the DSA measurement problems, triggered by the issues of the existing SA measurement techniques, which consist the core of this paper, and they are thoroughly discussed in the next sections. As a result, we move towards the need for a change in the paradigm and a more holistic viewpoint that encompasses mental, physical, and environmental aspects, far beyond individual and team concepts.

2 Previous Work

Stanton *et al.* [4] identified three approaches to describe the different contexts, in which the SA concept was developed and measured over the years. These approaches can be categorised as: (a) physiological, (b) engineering, and (c) ergonomics, and they were developed in parallel to social, technical, and socio-technical systems accordingly. In practice, the different SA measurement techniques are rooted in these three approaches that correspond to researchers' different perceptions of SA: individual, technical, or systemic endeavour.

The first approach perceives SA as an individual psychological phenomenon. It has gained the interest of many researchers, such as Endsley, who consider SA as a cognitive in-the-head process, without taking into account that human reasoning is usually affected by outer stimuli, owing to their communication with their environment, whether it consists of human or nonhuman elements. The second approach, i.e. the engineering one, describes the "world view" of SA [4]. In this approach, SA is considered to be affected mostly by information possession and flow, as well as by technical infrastructure, e.g. computers, displays, information systems etc. The way in which information is presented by artifacts influences SA by determining how much information can be acquired, how accurately it can be acquired, and to what degree it is compatible with SA needs [10]. The third approach is based on the idea that SA is distributed and it emerges from the interactions between human and nonhuman system elements, because the system is viewed as a whole. All in all, the DSA aspect combines the view of SA in the mind and SA in the world [4].

2.1 Types of SA Models

To explain the different aspects of SA, scientists have introduced seven SA models. These are: (1) individual [11, 12], (2) team and (3) shared SA [2, 3, 9], (4) collective SA [13], (5) meta-SA [2], (6) compatible SA [3], and (7) distributed/emergent SA [13, 14].

Individual SA is an individual's dynamic understanding of "what is going on" [1] around him/her. Team SA is usually examined in combination with shared SA. The latter is the common understanding of the situation, whereas the former is composed by team members' individual SA [4], along with their shared SA. Collective SA, on the other hand, is the sum of the individual SA, i.e. the SA that each team member bears, without having necessarily a common understanding over the situation, in which their environment finds itself. DSA is initiated by distributed cognition theory [15], according to

which cognition is achieved through the coordination among system elements. Researchers, who adopt the DSA model, embrace the notion of emergence, according to which a property is emergent when it cannot be detected on a single system element, but on the system in its wholeness. Compatible SA refers to elements that hold a distributed system together, and each element and agent in this system has his/her/its own awareness, related to personal goals and understanding over the situation. Compatible SA is not shared, since each team member's view on the situation is different, despite using the same information [16]. One can grasp the difference between compatible and shared SA, by keeping in mind that compatible SA is usually represented by puzzle pieces [3], while shared SA is the intersection between two or more sets. Moving a step forward, Salmon *et al.* [2] introduced the awareness of other agents' awareness, i.e. meta-SA stems from the fact that the knowledge of other agents' knowledge is contained in the system, such that each agent could potentially know where to go, when they need to find something out or manage a situation.

2.2 Existing SA Measurement Techniques

Stanton *et al.* [17] have reviewed more than thirty different SA measurement techniques, eighteen of which have been used by many scholars. In their paper, Stanton *et al.* [17] have categorised these measurement techniques into six general categories in terms of individual SA: (1) freeze probe techniques, (2) real-time probe techniques, (3) self-rating techniques, (4) observer rating techniques, (5) performance measures, (6) process indices, as well as into three categories, and shared SA: (1) team probe-recall techniques, (2) observer rating team SA, and (3) team task performance-based SA measurement techniques.

Table 1 lists the ten most extensively researched categories of SA measurement techniques, together with a brief description of the experimental measurement settings, under which they are executed. The corresponding weaknesses of each category, as being detected in the context of socio-technical systems, are presented in column C. From row 1 until row 6 the individual SA measurement techniques are listed. The team SA measurement techniques are included in rows 7* to 9* (the '*' symbol indicates team SA measurement technique). Finally, the DSA measurement technique is listed in row 10.

Table 1. Categories of SA measurement techniques, description, and weaknesses

	A - category	*B* - description	*C* - weaknesses
1	freeze probe e.g. SAGAT, SALSA	random activity freeze and questionnaire answering, based on agents' knowledge and understanding of the situation	- they constitute interference in the normal course of events within the system - observers measure what agents know, not how knowledge is obtained or maintained
2	real-time probe e.g. SPAM, SASHA	on-line and real-time, observers take into account both the content of responses and the time to respond	- rushed intensity of the respondents attention - observers may neglect other work packages, not currently examined - the bigger the system the greater the volume of observers (e.g. different mental models, cost concerns etc.)
3	self-rating e.g. SART, SARS	post-trial subjective measurement of agents' SA [5]	- individuals subjectively rate the quality of their own SA - responses affected by agent's psychological background - individuals do not rate their SA in each operational sub-process, they generally rate themselves, in terms of their overall performance
4	observer-rating e.g. SABARS	SMEs observe participants in action and score their performance via predefined "benchmarking" behaviour	- observers judge what is going on in individuals' heads from their "outer" attitude, over- or underestimation of individual SA - acceptable behaviour and high performance do not (necessarily) entail high SA - individuals know they are observed for specific behaviours, i.e. feigned good behaviour to avoid low scoring
5	performance measures	measuring relevant aspects of participants performance, e.g. 'kills' and 'hits' imply the success or the failure of the mission [5]	- unclear how the measured characteristic is linked to SA - good performance of one part of the system does not mirror the SA of the entire system - satisfactory SA does not mean that a sub-process will definitely run smoothly
6	process indices e.g. eye trackers and software	measurement of cognitive processes employed by participants to develop and maintain their SA	- eye-tracking machines rate the concentration of the human eye by perceiving data from behaviours considered to be related to SA - "look-but-failed-to-see" (Brown 2001), i.e. the individual 'looks' or 'sees' (e.g. eye-tracking devices grasp the motion of the human eye, without, however, being able to decide whether the observed agent comprehends the stimulus or just looks at it)
7*	team probe-recall	SA related questions are posed to all team members, one-by-one, during freezes in task performance	- difficult to be applied during real world collaborative tasks, i.e. used in simulated environments (Bolstad *et al.* 2005)
8*	observer-rating team SA	observers observe team performance and rate each individual team member about his/her SA as well as the shared awareness	- observer's measurement is subjective - it is not clear if observers measure individual, team, and/or shared SA
9*	team task performance e.g. CAST	examines responses to changes in team processes and environment, i.e. how aware the entire team and each individual, within the team, are	- roadblock scenarios work like preparedness exercises - unclear relation between performance and SA
10	DSA network-based approach	connections are important to explain SA in terms of collaborative systems. DSA is a set of information elements [3]	- networks only allow the representation of information flow between the interacting human and nonhuman agents - depiction not a measurement technique

2.3 A Network-Based Approach for Measuring DSA

Up to now, there is only one reported technique which claims to measure DSA. Indeed, Salmon *et al.* [2] and Stanton *et al.* [16], guided by the notion that there is a shift from models accounting for SA "in-the-head" [3] to models accounting for SA held by systems [18], attempted to introduce a theoretical approach as a measure of DSA under the perspective of complex socio-technical systems. Salmon *et al.* [3] also point out that it is crucial to describe the current situation of the examined system by using the available information, as well as taking into account who has the 'ownership' of that information, and how the different agents interact with each other via numerous technical system elements in order for the awareness to emerge.

However, the same authors acknowledge that there is an oxymoron in their technique, which is based on propositional networks. On the one hand, they introduce this technique as a network-based approach for measuring DSA, while on the other hand, they characterise this technique as a qualitative depiction of SA. According to Salmon *et al.* [3] (p.71) *"The propositional networks approach therefore differs from existing SA measurement approaches propositional networks do not attempt to quantitatively score each agent's SA quality they describe the content of the system's DSA during task performance and the usage of this information by the different agents involved."* Indeed, their propositional network approach to measure DSA is, in practice, a stepwise description and guidance for studying and depicting agents and networks of agents involved in the acquisition and maintenance of SA through information processing and assessment. The outcome of this method is qualitative and it mostly bears a resemblance to semantic networks.

3 Why It Is Not Worthy to Combine the Existing SA Measurement Techniques

To overcome the limitations of the existing SA measurement techniques, one might consider combining them. That would have probably been an acceptable strategy, if the examined system did not contain numerous elements and agents distributed at different hierarchical levels. However, in terms of measuring DSA in complex socio-technical systems, there seems to be some aspects of incompatibility. The first incompatibility refers to the measured objects. For instance, the real time probe techniques (Table 1 - row 2) measure individual's knowledge and understanding regarding what they see happening around them, whereas observers, in observer-rating techniques (Table 1 - row 4), identify and 'collect' benchmarking behaviours that are supposed to convey a positive SA-related conclusion. The second incompatibility refers to the experimental conditions of the measurements. Specifically, the existing measurement techniques are governed by mutually exclusive constraints regarding the processes and tools they integrate. An example of

mutually exclusive constraints exists between freeze (Table 1 - row 1) and real-time probe techniques (Table 1 - row 2). Typically, in freeze probe techniques, a task is randomly 'frozen' in a simulation of the task under analysis, all displays and screens are blanked, and a set of SA queries regarding the current situation at the time of the freeze is administered [7]. On the contrary, real-time techniques perform measurements on a real-time base while the participant is performing the task under analysis [7].

An example that combines both aspects of incompatibility includes the freeze probe, e.g. SAGAT, and self-rating, e.g. SART, SA measurement techniques. For Salmon*et al.* [7] specifically, SAGAT and SART techniques are entirely different, because the former queries participants for their knowledge of task-specific elements, whilst the latter does not refer to the specific elements related to the task, rather it focuses on generic, overall task characteristics. Generally, in freeze probe techniques on the one hand, experts interview agents [7] about their own view in relation to their understanding about the current situation of the system, whilst in self-rating techniques users make use of rating scales, which are more structured and somehow 'quantitative' compared to open-answer questions. This entails that one needs to modify the measurements, at least of one of the two techniques, to combine the results of the two measurement techniques in order for them to be comparable and to be able to 'draw' a joint conclusion about SA. Any kind of intervention in values may harm the genuineness of raw data, be time consuming, and/or lead to additional costs.

4 Issues on DSA Measurement

Among the literature of SA measurement techniques, there are reviews which identify most of the profound defects of the aforementioned techniques, e.g. time-consuming processes, training is presupposed, huge amount of resources is required etc [3, 18]. However, they fail to detect the deeper problems that underlie the lack of proper SA measurement techniques in complex socio-technical systems. In this section we group the most crucial points that render the given SA measurement techniques quite insufficient.

1. Unclear context and definition of system boundaries: SA measurement techniques were at first developed to measure individual SA in the field of aviation, e.g. freeze probe, real-time probe techniques etc. Researchers, who developed these measurement techniques, made their assumptions according to the conditions that exist in cockpits or in air traffic control towers, which are confined control rooms, under the notion that they do not keep up with the idea of extended and multi-agent socio-technical systems. What is more, researchers, such as Boy [19], insist that traditional SA-related army models are not enough to illustrate and/or comprehend the design and management of complex socio-technical systems, seeing that authority sharing is crucial for the distribution of awareness between system agents. Unfortunately, the conceptualization of SA in confined systems, within which agents

act in their own 'microclimate' in isolation, does not provide a realistic model of what is happening in complex socio-technical systems. Thus, the conclusion to be drawn from this claim is that, researchers need to set the system's boundaries and determine the level and depth to which they are going to investigate its operations, before adopting any kind of SA measurement technique. However, to do so, it is important to clearly define the roles, duties, and tasks of all agents who take part in the operation within a system. If for example, the observer is about to measure shared SA, he/she should consider both individual and team roles. Hence, the decision concerning the selection of the appropriate SA measurement technique depends on the assumptions that a scholar makes when determining the boundaries and the elements of the system.

2. SA models depict the individual's in-the-mind process, i.e. emphasis has been placed on individual level: The majority of SA approaches is based either on individual SA models, e.g. [1], or on team SA models, e.g. [3, 9]. Most of the widely known SA models, e.g. the three level model [1], the perceptual cycle [20] etc, only illustrate what is going on within one's own head, without taking into account his/her responses to events that possibly stem from interactions with other humans, artifacts, and/or environments, and it is inevitable to affect the inner operations within an individual's head. But the fact that a person can actually have high levels of awareness, relative to the current system's situation, does not entail the same levels of team SA. While trying to measure the SA on a system level, it is problematic to partially focus on the awareness of individuals, because this may lead to an incorrect evaluation, e.g. in case of team probe-recall techniques (Table 1 - row 7*). For example, some agents and/or elements may have low SA even when they have an efficient level of SA acting as team members. Things are getting more dubious when a researcher strives to depict and/or measure more complex types of SA, such as DSA. Hence, it is not sufficient to examine the individual focus of attention, but the system's focus as an entity.

3. 'Blurred' perception of what is going to be measured: None of the existing SA measurement techniques clarifies what characteristic and/or behaviour is about to be measured. Even when the theory behind the technique exemplifies what is going to be measured, the output of the measurement is different from the pursued objective, like it happens in the case of performance measures (Table 1 - cell 5B) owing to the unclear relationship between SA and performance [3]. In freeze probe techniques, as another example, human agents describe what they see happening around them, however, this does not entail that what they see is consistent to what really happens in the system. For instance, when a person's attention is captured by a situation, this is not necessarily equivalent to being (fully) aware of that situation. Process indices, where body actions are those observed (Table 1 - row 6), imply that when someone acts in an acceptable way, then he/she is aware of the situation. Researchers have developed this group of techniques making

the assumption that SA is what happening within one's head. It is thus an oxymoron to monitor body reactions, which may be biased, deliberate, or even a "by-the-book" attitude [3], and do not necessarily mirror the "in-the-head" cognitive, SA-related, operations (Table 1 - cell 6C). The incomplete problem statement may possibly lead to the disorientation of the solution. In a nutshell, the crucial deduction, on which researchers should put all their efforts, is the accurate and elaborate statement of the problem before making any attempt to solve it.

4. Information as the only factor that determines SA levels: Information flow is considered as the most significant factor that affects the SA formation [21]. The given models and measurement techniques focus on information flows and they are motivated by the notion that the more the information that enters the system, the more awareness the system will obtain. Researchers, for instance, make the assumption that individuals who possess much information perform better, and are aware of more elements in their environment, however, in complex collaborative systems, this is by far simplistic. What is needed in order to measure SA is not memory but comprehension, which is not necessarily proportional to available information. The linear connection between information and awareness is quite simplistic, since information requires, at first, filtering and further processing in order to be usable by the system. In complex socio-technical systems, information is not the only component that contributes to SA; that is, information triggers awareness, but does not entirely shape it. Thus, existing measurement techniques neglect to investigate the interactions between system agents and elements, and although this gap has already been detected by researchers, e.g. Salmon *et al.* [3], it is not bridged yet. Team probe-recall techniques (Table 1 - row 7*) is an illustrative example of disregarding such interactions. Considering that this technique serves to measure team SA, it seems incomplete to pose the same questions to all team members, individually, omitting to acquire information about the shared, collective, and/or compatible understanding of the situation.

5. Researchers apply SA measurement techniques when the system is already operating: None of the already known measurement techniques is applicable to the design phase of the system, as a precautionary measure for enhancing and preserving the awareness of system's possible future states. Some of them, for instance, require the freeze of operations, e.g. freeze probe, whilst others are performed in real time conditions, e.g. real-time probe, whether they comprise self- or hetero-measurement. Unfortunately, in none of the cases was awareness investigated from the design phase of the system. Engineers however, should be able, right from the early design phase, to design the system in such a way that it could operate in an efficient, effective, and safe way in respect to the scope and higher goals of the system, and carry at the same time those properties that are desirable and may empower awareness.

6. All SA measurement techniques arrive at qualitative conclusions: The existing SA measurement techniques give qualitative results, and because of this, they are subject to subjective collection and interpretation of data and information. In case of SART (Table 1 - row 3) self-rating technique, for example, respondents rate themselves on a scale from 1 to 7 when answering a typical question like this one: *"How familiar are you with the situation? Do you have a great deal of relevant experience (High) or is it a new situation (Low)?"* [7]. Nevertheless, this does not mean that the result is quantitative just because the answer has a numerical designation (e.g. from 1 to 7), but it is practically a numerical interpretation and an estimation of qualitative, e.g. cognitive, mental, emotional, characteristics that contribute to the shaping of SA under specific circumstances.

7. The means to implement measurement techniques: This issue is in close relation with the previous one, since it refers to the means of executing the existing SA measurement techniques. Namely, questions, e.g. questionnaires, rating scales etc, posed to individuals, limit the scope of SA and focus the interest on an individual's opinion and awareness. A direct consequence is the underestimation of the technical parts of the system and the loss of the related information. When using questioning, it is also crucial to choose the appropriate and understandable, according to mental models and experiences, wording and question formulation to avoid misunderstandings or divergence from the core inquiry, i.e. what does the question tries to elicit from the respondents. In addition, techniques where observes 'draw' the picture of the system, judging by what they see other people do, bear the risk of differently understanding the same situation. Specifically, observers, pursuant to their mental models, guess what other agents perceive and have in mind about the current system setting, regardless of the possible chasm between their mental models. A method structured in such a way, that is possible for the examiner and/or the examinee to misunderstand its initial goal or differently interpret its qualitative result, is not considered ideal for comparative analyses in engineering systems.

5 Conclusion

Judging from the literature, researchers contented themselves with sweeping generalities about the weaknesses of the existing SA measurement techniques, failing to think 'outside the box', i.e. they did not question the context of the SA approaches and measurement techniques. However, complex sociotechnical systems require more holistic reasoning and targeted approaches. This, in fact, justifies why neither traditional SA measurement techniques, i.e. individual and team ones, nor the DSA network-based approach [3] are entirely adequate and/or valid for the measurement of DSA in complex sociotechnical systems settings. For this reason, and by looking deeper into the fragility of the existing SA measurement techniques, we concluded to the

seven issues that emerge from the underlying need to change the paradigm. Hence, in order for researchers to avoid a dead-end technique for the measurement of DSA, they first need to explore the possibilities of resolving the above issues in the context of complex socio-technical systems.

Within the existing context (i.e. using the existing SA measurement techniques alone is not sufficient enough to obtain an estimate of DSA in complex socio-technical systems) and the current technological basis (i.e. we cannot constantly monitor human brain functions and reactions to stimuli), it is the cognitive and distributed 'character' of DSA that possibly renders its direct measurement quite a challenging task. Thus, we incline to conclude that the development of a DSA assessment approach, based on palpable and measurable system elements and behaviours, seems to be a feasible and realistic solution to the problem explained above. The word 'assessment' was intentionally chosen, since Oxford dictionary defines it as an opinion or a judgment about somebody/something that has been thought about very carefully, in contradiction to 'measurement', which is the act or the process of finding the size, quantity, or degree of something and seems to be extremely difficult, if not impossible, for DSA in the context of complex socio-technical systems. But, to deal with the challenge, whether it is possible to effectively assess DSA or not, this may probably presuppose a shift in the perception over DSA. Perhaps there is no sufficient solution so far, not because of researchers' inability to see, notice, or demonstrate the system's 'mechanisms' that lead to the emergence of DSA, but owing to the fact that they need to shift their viewpoint and 'update' their mental models in terms of SA and DSA, specifically. This, in turn, may be advantageous to take some preliminary steps to resolve issues surrounding a DSA assessment approach, and, as a previous step, to determine the impact of system elements on the DSA formation process, so as to redesign the existing system or to choose a more advantageous one, judging by the degree of DSA that emerges from it.

References

1. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Human Factors: The Journal of the Human Factors and Ergonomics Society 37(1), 32–64 (1995)
2. Salmon, P.M., Stanton, N.A., Walker, G.H., Baber, C., Jenkins, D.P., McMaster, R., Young, M.S.: What really is going on? Review of situation awareness models for individuals and teams. Theoretical Issues in Ergonomics Science 9(4), 297–323 (2008)
3. Salmon, P.M., Stanton, N.A., Walker, G.H., Jenkins, D.P.: Distributed situation awareness: Theory, measurement and application to teamwork. Ashgate Publishing, Ltd. (2009)
4. Stanton, N.A., Salmon, P.M., Walker, G.H., Jenkins, D.P.: Is situation awareness all in the mind? Theoretical Issues in Ergonomics Science 11(1-2), 29–40 (2010)

5. Naderpour, M., Lu, J., Zhang, G.: A situation risk awareness approach for process systems safety. Safety Science 64, 173–189 (2014)
6. Stanton, N.A., Young, M.S.: Giving ergonomics away? The application of ergonomics methods by novices. Applied Ergonomics 34(5), 479–490 (2003)
7. Salmon, P.M., Stanton, N.A., Walker, G.H., Jenkins, D., Ladva, D., Rafferty, L., Young, M.: Measuring Situation Awareness in complex systems: Comparison of measures study. International Journal of Industrial Ergonomics 39(3), 490–500 (2009)
8. Leveson, N.: Engineering a safer world: Systems thinking applied to safety. MIT Press, Cambridge (2011)
9. Salas, E., Prince, C., Baker, D.P., Shrestha, L.: Situation awareness in team performance: Implications for measurement and training. Human Factors: The Journal of the Human Factors and Ergonomics Society 37(1), 123–136 (1995)
10. Endsley, M., Jones, W.M.: Situation Awareness Information Dominance and Information Warfare. Logicon Technical Services Inc., Dayton (1997)
11. Endsley, M.R.: Design and evaluation for situation awareness enhancement. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, pp. 97–101. SAGE Publications (1988)
12. Sarter, N.B., Woods, D.D.: Situation awareness: A critical but ill-defined phenomenon. The International Journal of Aviation Psychology 1(1), 45–57 (1991)
13. Smart, P.R., Bahrami, A., Braines, D., McRae-Spencer, D., Yuan, J., Shadbolt, N.R.: Semantic technologies and enhanced situation awareness (2007), http://eprints.soton.ac.uk/264351/
14. Stanton, N.A., Stewart, R., Harris, D., Houghton, R.J., Baber, C., McMaster, R., Salmon, P.M., Hoyle, G., Walker, G.H., Young, M.S., Linsell, M., Dymott, R., Green, D.: Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology. Ergonomics 49(12-13), 1288–1311 (2006)
15. Hutchins, E.: Cognition in the Wild. MIT Press, Cambridge (1995)
16. Stanton, N.A., Salmon, P.M., Walker, G.H., Jenkins, D.: Genotype and phenotype schemata and their role in distributed situation awareness in collaborative systems. Theoretical Issues in Ergonomics Science 10(1), 43–68 (2009)
17. Stanton, N.A., Salmon, P.M., Walker, G.H., Baber, C., Jenkins, D.P.: Human factors methods: a practical guide for engineering and design. Ashgate Publishing, Ltd. (2005)
18. Stanton, N.A.: Representing distributed cognition in complex systems: how a submarine returns to periscope depth. Ergonomics (ahead-of-print), 1–16 (2013)
19. Boy, G.A.: Orchestrating situation awareness and authority in complex sociotechnical systems. In: Aiguier, M., Caseau, Y., Krob, D., Rauzy, A. (eds.) Complex Systems Design & Management, vol. 126, pp. 285–296. Springer, Heidelberg (2013)
20. Neisser, U.: Cognition and reality: Principles and implications of cognitive psychology. WH Freeman/Times Books/Henry Holt & Co. (1976)
21. Endsley, M.R., Jones, W.M.: A model of inter-and intrateam situation awareness: Implications for design, training and measurement. In: New Trends in Cooperative Activities: Understanding System Dynamics in Complex Environments. Human Factors and Ergonomics Society, Santa Monica (2001)

The Hidden Perils of Addressing Complexity with Formal Process – A Philosophical and Empirical Analysis

Paul Nugent and Emilio Collar Jr.

Abstract. This paper analyzes ethnographic interview data at two different points in time to assess the impact of formal process on the production process in a systems engineering context. While formal processes are adopted by organizations to grapple with increasing levels of complexity, they nonetheless make strong assumptions about the nature of work and the nature of the worker leading to unintended consequences. The fields of philosophy of technology and labor studies each pose deep questions about the nature of work and technology and how changes in them affect human experience and provide a compelling context in which to analyze the ethnographic data. We conclude that formal processes tend to ignore the social dynamics of the production process resulting in systemic quality problems. We also conclude that both the philosophy of technology and labor studies would benefit from a more social rather than individualistic framing of technology and production.

Keywords: Systems engineering, philosophy of technology, labor studies, technology, formal process, CMMI.

1 Introduction

In the summer of 2013 there was a formal engineering review of a major U.S. Navy program with the customer. In the morning of the second day of the review the customer walked to the front of the room, populated by close to one hundred engineers and managers, for his opening remarks. He had a grave and serious look on his face and he proceeded to say that he was not happy and that he was

Paul Nugent · Emilio Collar Jr.
Western Connecticut State University, Danbury, Connecticut, USA
e-mail: {nugentp,collare}@wcsu.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_9

unable to sleep the night before and that in all of his years on the program he had never seen so may instances where the presentations lacked polish, or that the technical/presentation had so many minor errors, and that on a whole he felt a widespread lack of commitment to "his program." He said that the reason why this contracting organization had been chosen year after year to support his programs is because of its high level of commitment and technical knowledge and that he was afraid it was in jeopardy.

The review continued and although the customer expressed some improvement in the content, he concluded by saying that it was still not up to par with our usual performance and that there would need to be another technical review known as a "re-review." The next week there was a special meeting convened with all of the review participants where program/technical management referred to the problem as "death by a thousand paper cuts." The technical lead rebuked the team for "letting him down." The stage was set for all of the presentations in the re-review to be given intensive scrutiny by upper level management, technical leaders, and the chief engineer. The re-review was conducted months later and the customer was very pleased with the results.

In complex organizations there are many possible contributors to performance failures such as this one. However given the systematic nature of the failure it is reasonable to suspect that the cause itself may be systemic – that is, a change in the organizational system that had occurred over the previous years. In particular, based upon analysis of the ethnographic data, while the introduction of new engineering processes appears to be a win-win scenario for all involved, a closer look at the basic assumptions of these processes reveals why they may have unintended consequences on productivity and quality. Specifically the analysis in this paper indicates that they ignore the social facets of the production process in complex environments and the fact that their introduction may, over time, undermine the effectiveness of these social processes.

In this paper we analyze ethnographic data drawn from a large systems engineering organization both at the time that the new process was introduced and then five years later to establish how the engineers themselves experienced the shifts that occurred in the work environment. A brief overview of the philosophy of technology and labor studies is presented to establish a deeper scholarly context within which to refine a research question as well as to think about the strong assumptions that formal processes (as technologies) make about the production process itself and the role of human beings.

2 Complexity and the Rise of Formal Quality Control Processes

A system is complex if it has many different interacting elements and complexity itself has been defined as the "study of the phenomena which emerge from a collection of interacting objects."[1, p. 3] While natural systems possess their own

connected parts and levels of complexity, human collectivities create connections of a social nature introducing almost limitless complexity. Human forms of organization have exhibited a steady increase in complexity as they have moved from agricultural, to industrial, to information age paradigms.

From a cybernetic perspective, with increasing complexity comes an even greater tendency toward entropy or chaos and the need, therefore, for systems of messaging and feedback to keep these *under control* [2]. Historically new technologies and processes have evolved to control the emerging problems of complex social organization and labor. For example, early accounting methods and the bureaucratic structure, in particular, are often credited with laying the foundation for modern institutions and ways of organizing [3][4]. In addition early organizational theory was focused on methods of managerial control and coordination as well as optimal "spans of control" within a hierarchy [5][6] as well as how to manage the uncertainty inherent in complex socio-technical environments [7].

However, in modern organizations, the control of *quality* (i.e., "quality control") has emerged as the dominant means by which to confront these high levels of social and technical complexity. While early quality control movements focused on the weeding-out of non-conforming parts through inspection practices, there has been throughout the twentieth century a shift "upstream" toward higher-level mechanical and administrative processes. Statistical Process Control (SPC), quality circles, Six Sigma, etc. focus on understanding the natural expected ("common cause") variations in these processes and the detection of abnormal ("special cause") variations that need to be identified and corrected [8].

These quality control programs were very successful in the production of relatively simple physical commodities or services and therefore were the natural choice for application to ever more complex production processes. For example, in large organizations of the 1970s and 1980s the volume of software in the internal functions of the organization or in products for personal consumption was growing exponentially and yet managers of software development activities equated this process to "trying to nail jelly to a tree." Tools and processes did emerge, however, that brought the complexity of software development under some level of control. Software debugging programs, higher-level programming languages, structured testing environments, automatic code generation, etc. aided the developer in bridging the gap between product conceptualization, implementation, and integration. Yet these still fell short of controlling the highly complex software development process itself.

As software production became more routinized it was still characterized by errors (bugs) and also by high levels of variation in the skills and competencies of the development teams. Management sought consistency and predictability of output and therefore the quality control (Statistical Process Control) paradigm became an already available and successful model to apply to this new control problem. Although each software product is different from others because each performs unique tasks, the new quality programs assumed that they should be similar to each other at least with respect to the ratio of defects encountered for a number of lines of code and level of complexity. The Capability Maturity Model

(CMM), in particular, was a process model of this type that assumes that at various points in development the "product" should be assumed to be defect-free and would only require inspection through "gate reviews" or "peer reviews" to identify defects [9]. More recently these process philosophies have evolved to ascend to higher levels of the organization to control the quality of products such as requirements specifications or system design documents at the "system level" in the name of Capability Maturity Model Integration (CMMI)[10].

This paper explores questions about comprehensive formal processes such as CMMI and attempts to situate them philosophically, sociologically, and empirically. For example, there is a relatively mature literature on the philosophy of technology and it is interesting to ask if formal process is a "technology" or is a special *form* of technology. There is also the better known stream of labor process research inspired by Karl Marx that asks how changes in organizational structure and technology (of which formal process introduction is a special case) affect the *experience* of work (e.g., alienation, deskilling) and the worker's control over the work. It is to these general lines of inquiry that we now turn.

3 Formal Process as Technology

Martin Heidegger's essay *The Question Concerning Technology* continues to be the most influential work on the philosophy of technology. In it he agrees that technology is both a "means to an end" and a "human activity," but also that it is not value-neutral and he asserts, "the essence of technology is by no means anything technological" [11, p.4]. By this he means that technology itself has the potential to reveal or conceal how man conceives of himself and things in his world as *beings*. While the early Greeks utilized technologies that, he claims, preserved a respect for the intrinsic nature of beings in the world (landscapes, animals, plants, people), modern technology transforms these *beings* into merely identifiable resources "at-hand" for man to put to use. In so doing, man "enframes" or comes to see things in his world as primarily resources and this, paradoxically, transforms man himself into a resource ("human resource") to be used within a social system while also convincing himself that he is master over, and in domination of, the natural order.

More recently philosophers of technology have criticized Heidegger for being too general and overly romantic in his interpretations of ancient Greek technology. In contrast, they focus on specific technologies and on the manner in which they enhance man's capabilities and in so doing influence the ways in which man experiences the world [12][13]. For example the telescope or microscope extends the scales that the biological human eye can grasp. The technology, in addition to augmenting man's abilities also changes how he approaches his world by shifting certain contents to the foreground while relegating all else to the background or periphery. Other scholars have pointed out that the philosophy of technology has focused almost exclusively on the capabilities that technologies provide while

ignoring the need for technologies to also secure systems from complex threat environments [14].

In this context, we may ask, is a formal process such as CMMI a technology? If so, does it square with the kinds of technologies that Heidegger and others have considered, or does it beg for a new philosophical treatment? On the one hand, a process such as CMMI is a means to an end and is a human activity, but it is not usually referred to as a technology. This may be due to the fact that they are not physical machines inspired by scientific discoveries but are rather formal rules governing the structure and sequence of human activities over time as well as a means to evaluate and control them. How, then, does CMMI alter or constitute the workers relationship to the product being developed?

4 Formal Process and Labor Study Literature

While Karl Marx is best known for his economic ideas about social class, false consciousness, and capitalist exploitation of labor, he was also a philosopher and sociologist who believed that social consciousness itself arises from the ways in which work is actually (materially) performed [15]. His assumption was that there are basic levels of freedom, self-determination, and control over production that properly ground man in relation to nature and in relation to a social community. Any labor arrangement that blocks these outcomes alienates man from his "true" nature.

Throughout the industrial revolution many studies have shown strong evidence that the introduction of new machinery and automation have the alienating effects that Marx outlined [16][17][18][19][20]. However other studies have shown that with automation there are other forces at play that can be argued to re-skill the worker and provide them with newer understandings of the process – in effect having the potential to reverse the alienating effects. For example in the ethnography *In the Age of the Smart Machine* Shoshona Zuboff provides ample empirical evidence that while automation of factory processes buffers the workers from a more direct, tactile, and sensory understanding of a process, it nonetheless *informs* them ("informated" them) about a broader and more systemic "big picture" of the production process [21]. Another study showed how the introduction of new technical equipment in a hospital setting became an opportunity for the workers to conceive of new organizational roles, relationships, and structures [22]. Therefore rather than assume that all automation deskills and "dumbs down" work, it is better to think about how the introduction of automation or new technologies change the worker's experience and work relationships.

It is in this vein that we look at formal processes such as CMMI to investigate how, if at all, it shifts the experience of the work and production relations.

5 Research Question

Both the philosophy of technology and labor studies take seriously the notion that the introduction of new technologies is important to man and his sense of self-worth, freedom, and his relationships with others. However it is also clear that neither field has a strong consensus over new technology's effects on the human experience and the human condition.

The goal of this research is not to lay this question to rest. Rather, it is the goal of this study to add another data point to these dialogues by inquiring as to how the introduction of a kind of technology (the CMMI formal process) alters the engineer's experience of the work and/or alters the social relations of production.

6 Research Method

The data for this study were drawn from two phases of a participant observation research undertaken between the 2005 and 2011 at a large defense contracting company in a small New England city. The company of approximately 1000 managers, engineers, and technicians designs and assembles launch control systems for Trident II submarines and guidance systems for nuclear missiles. Over 70 percent of the employees are professionals (engineers and managers). Work direction is organized in a matrix structure in which program managers oversee project cost, schedule and budget while functional managers are responsible for maintaining a pool of skilled/experienced workers to execute contracts. The systems they design are integrated into larger systems requiring extensive coordination with other organizational units, with customer (Navy) organizations, and other government contractors.

Eight interviews during the first phase (May, 2005) and ten interviews from the second phase (July, 2011) were performed. Each interview lasted approximately one hour and was recorded and transcribed. Ethnographic interviewing methods were used [23]. The interviews began with "grand tour" questions such as "Can you tell me about when you came to the company and the different positions you have held?" and "Please explain what your current role is and whom you work with." Probes were used to encourage the interviewee to elaborate on points or for clarification. Subsequent questions focused upon direct experiences with tasks and how CMMI affected them. Questions also centered on the nature of the work interfaces and the discrete interactions required by the work.

Grounded theory methods [24] were used to evolve codes and to develop theory. The early coding phase was intentionally flexible and reflexive – allowing the data to stimulate thinking about theoretical categories and relationships while also allowing the emerging themes to focus the subsequent data gathering. This phase elaborated on categories of interpersonal involvement in the work activities and the effects of CMMI on concrete tasks.

7 Analysis

The analysis is organized into the initial data gathering phase in which interviews were conducted while the new process (CMMI) was being introduced and then five-to-six years later when the process had been fully implemented and was now part of the taken-for-granted organizational culture.

7.1 Initial Reactions to Process Introduction

The initial reactions to the introduction of the new processes centered on an optimism that, although it requires more "overhead" and creation of "artifacts," it was nonetheless something that would help formalize what was previously more ad-hoc and therefore could be beneficial to the organization. According to one veteran systems engineer,

> I think [pause] CMMI is good. It's forcing people to do what they do in a more formal way and to basically document what they do, archive what they do, so that it's much easier to go back six months down the road and really see what happened. Um, I think this annoys a lot of people because they feel like they just can't do what they do. Now they have to spend more *time* doing what they do. I think the general opinion is that they believe it takes a lot more time to do things within CMMI than it was previously. I think it's just something that people gotta get accustomed to that, yes, when you work you need to have evidence that things are happening and document these things so that you can go back and see why certain decisions were made. And I think that's pretty much a lot to do with CMMI and also more formally directs doing tradeoffs versus people relying on gut feelings for why they chose a particular design. Basically CMMI just seems to formalize a process that we *have* been doing pretty much across the board.

However there were some systems engineers who sensed that the process focus could shift the worker's attention away from more important goals. One of the more technically respected and quickly rising systems engineers stated:

> I would say that the drive for the artifact could compromise the quality of the product because you spend more of your normal working day producing artifacts rather than developing the system that would really work. And while you need a certain amount of best practice, there's a line. You can have too much of a good thing. You can have control for control's sake rather than having tools in place to enable engineers to work together more effectively.

In conclusion, the first phase of this analysis shows that engineers were not concerned that the introduction of the formal process would change how the work was done or how it might influence the distribution of technical knowledge and responsibility within the organization. Rather, there was a belief that it would require additional overhead work to capture metrics and that it might also shift attention away from the most important goals.

7.2 Five Years Later: A Deeper Look at CMMI Process

Two main themes emerged from the analysis of the second data set. The first theme identifies how the introduction of CMMI created a shift in technical understanding/knowledge and role boundaries. The second theme represents a shift in accountability/responsibility away from the individual and toward the formal process.

Within this organization there had always been a strong norm that for any technical role it is the individual worker's responsibility to be autonomous and to discover for themselves, through interactions with others and hard work, the true nature of their task. Claims of "that's not my job" or "you're encroaching on my turf" were rare and those who succeeded did so because they were able to quickly transcend their formally prescribed role and demonstrate that the critical goals of the organization were forefront in their minds. However throughout the years in which CMMI was being assimilated there was a shift in this norm. According to a senior systems engineer:

> There was a lot of cases back on some of the other programs where you had a software designer that, the only way he could write the software, because the spec. doesn't go down to the pimple on the gnat's butt like they are today for software requirements – he would have to deal with a level of *interpretation*, and they'd go back and talk to the spec. writer and it would all get interpreted eventually, and by the time you got done, if someone went back and asked that software designer, 'how does this work?', they could give you an answer almost as good as the systems engineer. I'm not sure you can do that today.

The tightening of engineering processes means that those performing work that previously required more technical and social knowledge are now provided with, and therefore only need to seek out, knowledge within more narrowly prescribed boundaries. Again, according to the senior systems engineer:

> I think part of the idea here is that you only have a limited number of people who really understand how the system is designed and then any time you get done with a software requirements spec., anybody can code it. OK. That wasn't the case [before]. You really needed someone with

> a bigger familiarity of the system in order to do that. I don't know if that's a good thing or a bad thing, but I think it's different.

Therefore with the formalizing of the production process there is also a clarification and reification of formal roles. At any level the formal process more clearly identifies roles and the interdependencies between roles that discourages transcending these boundaries to obtain broader or more general understandings of complex technical and social environments.

In addition to a shift in domains of knowledge, the second theme of the analysis shows a shift in accountability. According to one systems engineer:

> If you have an artifact to produce and regardless of where your true commitment falls, when somebody draws a line in the sand and says you have to get something done by a particular point in time and then there's something else that *really* should be done by a particular point in time but doesn't directly reflect on that artifact, where do you place the priority? You prioritize based on what you're being measured to when push comes to shove. If what you really needed to do was take the time to work with somebody else to work out a critical interface, … let me back up. It promotes more self-interest. If individuals are being measured with milestones rather than measuring the *crew*, you know, based on what they really accomplish, people are going to *act* individually.

Finally, while this shift in accountability from the group to the individual was encouraged by the metrics gathering structure there was a parallel shift in which, in framing the individual worker as a measurable role occupant, it was now the "process" (technology) that was primarily responsible for the overall quality of the product. In other words, whether the individual product was deemed a success or a failure, it was the process (organization) that claimed credit or blame because it was the process that, at least formally, provided the structures and rules for matching resources (engineers) to specific tasks (e.g., requirements specifications and design documents) and monitoring for shifts in quality. This change, perhaps more than any of the others identified above, was likely to have contributed to the customer's dissatisfaction in the introductory vignette because from it evolved a tendency where engineers were less personally committed to their product and were routinely reassigned from one product to another as an available "resource." Through time this transformed the culture to one of less commitment to organizational outcomes and less identification with one's product for success or failure as these outcomes are now attributed to the process/technology rather than to the human being.

8 Conclusion

The foregoing analysis provides several themes that would explain how the social dynamics largely responsible for the quality of systems engineering products were likely to have shifted due to the implementation of CMMI and resulted in quality problems. Beyond this practical conclusion, the analysis asks us to think about how the introduction of formal process (e.g., CMMI) has implications for philosophy of technology as well as for labor studies and it is to these that we will now turn our attention.

8.1 Philosophy of Technology

Heidegger was most interested in ways in which technology can either reveal or conceal the "truth" (*aletheia*) of being. However, in this study the products that were being produced and reviewed in peer reviews were representations (requirements specifications and design documents) of *systems-to-be*. Therefore *being* in this context is a future state of affairs – the system, and the copies of it, that will exist on various Navy platforms and laboratory sites. However, while Heidegger prefers to focus on the psychological effects of technology on concealing being as mere *resources* rather than as *things-in-themselves*, the analysis compels us to shift this preference about truth to something more social. What the thing is *to be* is not something that is an already existing being that is concealed through the technologies/processes, but rather it is something that is a product of social interaction and dialogue – in particular the negotiated compromises between often conflicting customer and managerial expectations that are settled in the review process.

Therefore if we think of CMMI as a technology that alters this social process, we are coaxed into considering the role of *understanding*. Understanding is not merely a comprehension of engineering methodologies nor of the objective functionalities of the system components, but is also, critically, a comprehension of social information and where it is located. Truth, then, is something that is not a psychological act of defining and labeling, but rather is something that is a social process of assessing what is *to be* in the light of multiple and potentially conflicting expectations by powerful stakeholders.

From the analysis, formal engineering process (e.g., CMMI), if it is to be viewed as a technology, appears to attenuate the breadth of this understanding as its goal is to compartmentalize or modularize technical knowledge. In other words, it assumes that tasks are delineated so that the broader social understandings are not necessary to determine the future system and instead the producer is furnished by design, in a perfect world, with all of the information required to perform the task.

Therefore this is an invitation to relax Heidegger's more individual/psychological view of technology's effects to also include a more complex sociological conceptualization of "truth" as a *social reality* that relies on

establishing understandings and which is influenced to some degree by the degree of formalization (e.g., CMMI). Furthermore, considering more microscopic themes in the philosophy of technology such as Ihde's, we again are forced to consider more seriously the social nature of experience and *capabilities* that are not individual skill, knowledge, and perception, but in these work contexts are predominantly social understandings to determine the truth of what the future system needs to be. In this light, the introduction of formal process such as CMMI, then, has the effect of reducing or blocking social understandings and identities because it is premised on the orderly compartmentalization of "technical" knowledge while deemphasizing or denying the importance of broader social understandings of the expectations of key stakeholders.

8.2 Labor Studies

As with the philosophy of technology we conclude that labor studies also suffers from a bias toward the individual and the psychological and away from the social. Even in Zuboff's study of mill workers and the ways in which automation "informates" them to facilitate broader understandings of the production process, these remains understandings of a mostly non-social nature. Marx wrote extensively about the social impacts of capitalism and how it alienates men from one another, but he tended to focus on class consciousness and the connection between man and his own productive capacities. Instead, we see in systems engineering, a production process that is intrinsically social and wholly depends upon social understandings (multiple and conflicting expectations) as well as social interaction (negotiations and compromises in technical review activities). If we redefine "labor" as something with this significant social component, then we are in a position to reflect upon the analysis and how the introduction of formal processes such as CMMI may influence the labor process.

Based on the analysis, we see not only that the introduction of CMMI reduced the breadth of social understanding/knowledge but that this also was accompanied by a shift in social relationships. What was previously a strong culture of social commitment and accountability to coworkers and the broader collective endeavor (program) became, instead, one where credit and blame were shifted or reallocated the process or to management as a whole. According to this premise if an individual produced an inferior product it was because the process failed to train him/her or failed to identify the appropriate skills/knowledge "fit" for the task rather than because the engineer performed poorly. Similarly, where the product is deemed to be superior, it is the process "owners" who claim the majority of the credit. This, then, suggests that we consider the idea of alienation from a more sociological point of view where the *meaningfulness* of work derives primarily from one's social identification with one's product and how it is perceived (admired/critiqued) by coworkers and that in this context the introduction of formal processes such as CMMI reduces this identification and the social status that it affords.

8.3 Future Directions

This paper provides an explanation that one major instance of an organizational failure was due to the introduction of a formal process representing very strong assumptions about quality and human roles in production processes. It also contributes to and expands the fields of philosophy and technology and labor studies by recommending that they include more social framings of technology and production relations. Together these encourage us to take a fresher look at "process" in general and how, as a system of time-sequenced rules, it is able to influence the psychology and sociology of work. The review activity, in particular, as a social activity that is fundamental to formal processes such as CMMI in complex environments, deserves closer examination.

References

1. Johnson, N.F.: Simply complexity: A clear guide to complexity theory. Oneworld Publications (2009)
2. Wiener, N.: The Human Use of Human Beings: Cybernetics and Society. Da Capo Series in Science (1950)
3. Weber, M.: Bureaucracy. In: Shafritz, J.M., Ott, J.S. (eds.) Classics of Organization Theory, 3rd edn. Brooks/Cole Publishing Co., CA (1973)
4. Perrow, C.: Complex Organizations: A Critical Essay, 3rd edn. McGraw-Hill, Inc., New York (1986)
5. Shafritz, J.M., Ott, J.S. (eds.): Classics of Organization Theory, 3rd edn. Brooks/Cole Publishing Co., CA (1973)
6. Arrow, K.: The Limits of Organization. W.W. Norton & Company, Inc., New York (1974)
7. March, J., Simon, H.: Organizations. John Wiley & Sons, Inc. (1958)
8. Juran, J.M.: The Quality Trilogy, Quality Progress (August 1986)
9. CMMI Product Team: Capability Maturity Model Integration (CMMI) Version 1.1. Carnegie Mellon Software Engineering Institute (2002)
10. Chrissis, M.B., Konrad, M., Schrum, S.: CMMI for Development: Guidelines for Process Integration and Product Improvement, 3rd edn. Addison-Wesley (2011)
11. Heidegger, M.: The Question Concerning Technology. In The Question Concerning Technology and Other Essays. Harper & Row Publishers (1977)
12. Ihde, D.: Philosophy of Technology: An Introduction. Paragon House Publishers, New York (1993)
13. Ihde, D.: Heidegger's Technologies: Postphenomenological Perspectives. Fordham University Press, New York (2010)
14. Nugent, P., Ali, A.: Frankenstein's other Monster: Toward a Philosophy of Information Security. International Journal of Computer Science and Information Security 10(4) (2012)
15. Marx, K.: Karl Marx, Selected Writings in Sociology & Social Philosophy. Translated by T. B. Bottomore. McGraw-Hill, New York (1956)
16. Blauner, R.: Alienation and Freedom. University of Chicago Press, Chicago (1964)

17. Braverman, H.: Labor and Monopoly Capital. Monthly Review Press, New York (1974)
18. Burawoy, M.: Manufacturing Consent. The University of Chicago Press, Chicago (1979)
19. Clawson, D.: Bureaucracy and the Labor Process. Monthly Review Press, New York (1980)
20. Edwards, R.: Contested Terrain. Basic Books, New York (1979)
21. Zuboff, S.: In the Age of the Smart Machine. Basic Books (1988)
22. Barley, S.R.: Technicians in the Workplace: Ethnographic Evidence for Bringing Work into Organization Studies. Administrative Science Quarterly 41, 404–441 (1996)
23. Spradley, J.P.: The Ethnographic Interview. Harcourt Brace Javonovich College Publishers, Fort Worth (1979)
24. Strauss, A., Corbin, J.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage Publications (1990)

A Formal Foundation of Systems Engineering

Dominique Luzeaux

Abstract. In this paper we discuss a formal foundation of systems engineering based on category theory. The main difference with other categorical approaches is the choice of the structure of the base category (symmetric monoidal or compact closed) which is, on the one hand, much better adapted to current modeling tools and languages (e.g. SysML), and on the other hand is canonically associated to a logic (linear logic or fragments thereof) that fits better with systems engineering. Since that logic has also a rich proof theory, this allows us to propose a global formal framework that encompasses: system modeling, system specification, and property verification.

1 Introduction

Systems engineering is a rich technical domain where many specific formal models have been developed in the last decades. In order to understand the scope and limit of current work, we will distinguish three main subdomains within systems engineering: system modeling, system specification, property verification. This decomposition follows more or less the partitioning of information processing or control theory, where the denominations would be modeling, analysis, synthesis.

System *modeling* is a descriptive preoccupation related to system theory and is a key element of the upper stream part of the traditional systems engineering cycle. System *specification* focuses on the left leg of the systems engineering cycle, while property *verification* is the formal part of the right leg.

Although these three domains overlap the main engineering activities (excluding development and manufacturing), the corresponding formal theories are different since most work in the literature starts from different a priori assumptions in each domain. For instance, property verification through model checking is mostly defined in the following way: the system is known through its behaviors, i.e. the sequence in time of its inputs and outputs, and logical properties are checked for

Dominique Luzeaux
e-mail: dominique.luzeaux@polytechnique.org

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_10

on their exhaustive exploration. Deductive verification works differently by proving theorems generated in some way from the system and its specifications. Both types of formal verification assume a logical theory is given in order to have a proof theory. On the other hand, system modeling is usually done by introducing a formal system that focuses either on the structural features of the system (its decomposition into subsystems and the relations between them) or the functional or computational features (the system is seen as a function transforming inputs and outputs, or as a process relating inputs and outputs). Although behaviors can obviously be generated by the formal systems, the formal models are not necessarily optimal for dealing simultaneously with both modeling and specification approaches. To sum up, system modeling relies usually on an intensional slant whereas property verification relies on extensional descriptions.

However, mathematically, there are relationships (adjunctions more than isomorphisms, i.e. one instance of a formalism can be translated into an instance of the other formalism, and conversely, and there is a canonical way to do both translations, but they are not necessarily reciprocal) between many of these various formalisms. A goal of this paper is to address such issues in order to provide Ariadne's thread through the maze of various ad hoc models.

2 What Is Necessary for an Adequate Formalization of Systems Engineering?

The various questions to address are: What are the key features of a system? How can the structure of the system be described? How is it possible to reason on the system? How can requirements on the system be expressed? How can requirements or properties of the system be mapped on the structure of the system, and conversely how does the structure of the system explain some of its properties during utilization?

A system is a "combination of interacting elements organized to achieve one or more stated purposes" (as defined for instance in the ISO/IEC 15288 standard). A formal model should therefore be able to distinguish the *elements* as well as the *relations* between them. These relations correspond to the organization and to the interaction of the elements. In the systems encountered in real world, there are both products realized by hardware, and services which are immaterial activities that can be composed. Therefore relations between objects and *composition* are the first key ingredients to model. At this point it should be noted that composition is either *sequential* (the first system's outputs are inputs of the second system) or *parallel* (two systems define a new system with inputs the inputs of both systems, and outputs the outputs of both systems).

The interactions between the elements involve three main kinds of flows: matter, energy, information. Depending on the type of flow, some resources can be *reused* (such as information), while others are *consumed* (matter, energy). This distinction should be also seen as a basic feature to model, if one wants to be able

to model large-scale complex systems that include software and hardware. It is all the more important with the increasing ubiquity of services in complex systems.

System modeling relies on decomposition and aggregation of systems. This means that systems, when composed together yield systems again. It should be emphasized that no specific algebraic structure on that aggregation is imposed (typically we do not need consider a Cartesian product structure, which imposes some unicity properties that are mathematically interesting but constrain actually the formal theories, and are not necessarily needed in the real world). Systems can also interact dynamically together, going beyond a simple bidirectional information exchange, through *feedback*: some of the outputs of the first system are fed as inputs to the second system, and conversely. Modeling feedbacks is therefore another key ingredient, and should not be misunderstood as mere composition of systems.

Providing a formal model that is generated by these three main ingredients will be the focus of the next section.

Reasoning about a system or a type of systems, in order to discuss property specification or verification, necessitates being able to express properties within a logical theory that characterizes exactly the systems considered, with a stress on the adverb "exactly". This may seem obscure, but the idea is that the means to express a property about a system should have the same power of expressivity as the means to describe a system. Indeed, if you are able to express properties in a language but cannot differentiate two systems that satisfy and do not satisfy the given properties, or if you can describe various types of systems but are not able to state properties relevant to ever single type, something is wrong. This adequacy between the formal tools we propose for the different systems engineering activities (system modeling, system specification, system verification) is the main contribution of our work, which qualifies hence as a formal foundation of systems engineering as a whole.

2.1 Informal Presentation of the Formal Foundations

We will use the mathematical framework proposed by category theory [29, 30] since, among other things, it provides a common language and a set of unifying concepts based on a relational approach. Furthermore that framework is strongly related to logic, with many established results characterizing the relationship between categories and logic [31].

A cornerstone of category theory is the systematic adoption of a relational viewpoint: everything may be defined as an arrow (a relationship or a transformation, or, more precisely, a morphism) between objects, and these objects themselves may be described by exclusive use of arrows (the preconceived conceptual difference between objects and arrows is simply that arrows can be combined with other arrows, and therefore have a "from' and a "to", but objects may also be defined as arrows for which the "from" and "to" coincide).

Here, the contribution to systems theory is clear: the systemic paradigm and category theory meet in the pre-eminence they accord to relational ontology, on both a static and a dynamic level, with the establishment of the evolution of relational structures. This is, moreover, where we find the main mathematical difference between category theory and set theory. While the latter characterizes mathematical objects by describing their internal structure, i.e. by separating them into different parts and elements, the former takes the opposite point of view and characterizes an object by its connections to other objects. Category theory is, moreover, used today as much in theoretical computing [16, 17, 18] (modeling information systems, databases, multiprocessor languages [7, 34] or providing a practical means of dialogue between specifications and implementations [11]) as in quantum physics, where it is used in modeling non-classical worlds [12] as well as in attempts to define a unified framework of physic and computation [21, 32].

Such a relational vision has two consequences. Firstly, an object is only described up to isomorphism, i.e. independently of a particular implementation. Secondly, the strictly representational aspects of an object are removed in favor of the structure of inter-object relationships, a process which reminds us of the key notions of architecture in systems engineering.

Moreover, one aim of category theory is to provide a precise definition of "naturality", which is used, among other things, to define natural transformations: they model the intuitive idea that complicated things may be transformed into other complicated things if we modify the corresponding sub-structures correctly and if the transformation is not too ad hoc in the sense that it may be defined by a general mechanism applying to every object being considered (the mechanism transforms locally a relational structure into another, such that the local pictures fit together into a global picture).

Another advantage of category theory is its relation with logic: it is possible to associate a logical theory to a given category, and conversely to a logical theory it is possible to associate a categorical model. Characterization theorems allow relating exactly some types of categories with types of logical theories: in other words, if some properties are added to the categories, then the associated logics have also specific properties, and conversely. E.g. Cartesian closed categories correspond to untyped lambda-calculus, toposes correspond to intuitionistic higher-order logic, well-pointed toposes to classical logic, compact closed categories to multiplicative linear logic, etc. This will of course be a keystone of our formal foundation since it provides the unifying link between system description (category theory), specification (logical theory) and verification (proof theory).

In the last years, description languages such as UML and more recently SysML have been developed to deal with specification and design of complex systems. If we look at the graphical meta-language of such languages, the structures look familiar: boxes with wires coming in and out, boxes within boxes… Indeed, all these diagrams can be obtained recursively from three main operations on boxes: sequential composition, parallel composition and feedbacks. However in order to manage all kinds of multi-input multi-output situations, and to introduce new boxes or wires wherever needed, one should also have additional operators on wires (see Fig. 1), which create arrows, regroup them or make them cross [19, 20].

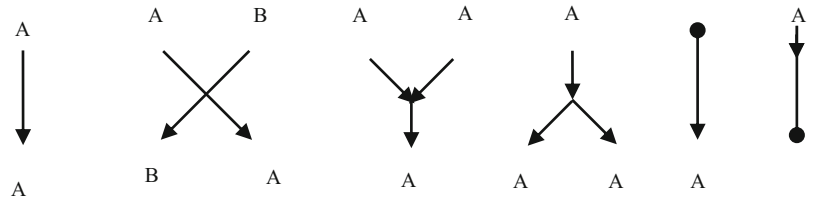


Fig. 1 Left to right: identity; transposition; identification; duplication; creation; deletion

By applying recursively all these operators, we obtain a graphical language able to represent systems with their interconnections, as well as combinations of systems and incoming (resp. generated) events, i.e. inputs (resp. outputs) coming from (resp. to) the environment and connected to the systems. This graphical language is the one underlying SysML (in this paper all arrows will be of the same type, but by enriching the base category with different types of morphisms it is straightforward to cope with that feature of SysML).

An advantage of category theory is that these operators can be interpreted as axioms, the combination of which defines a well-known class of categories – compact closed or *-autonomous categories: used in circuit theory [24, 25] and increasingly used in theoretical quantum physics [9, 21, 39] –, to which a special logic (a segment of linear logic) can be associated, as hinted in the previous paragraph. The beauty of this is that the aforementioned logic has been used independently to model and specify software systems [2], Web services [36], etc.

2.2 A Gentle Introduction to Category Theory and Categorical Logic

Category Theory. We shall now provide a rapid overview of certain basic concepts involved in this theory and provide illustrations of its contribution to the modeling of complex systems. A *category* is specified by a class of *objects* and a class of *arrows*, also known as *morphisms*. An identity arrow exists for each object, and for each arrow at the end of which a second arrow begins, we can define the way in which these two arrows fit together. Moreover, three arrows which may be put together do this in an associative manner, i.e. the first two may be put together with the third, or the first may be combined to the last two. A category may therefore be seen as an oriented graph, where the vertices are objects and the edges are the arrows. By applying definitions recursively, we can consider a particular category of which the objects are themselves categories: in this case, the morphisms between these composite objects are known as *functors*. More precisely a functor between categories maps objects of the first category to objects of the second category, so that a morphism between two objects of this first category is mapped onto a morphism between object images. Thus, a functor is a transformation which preserves, in a certain way, the base structure between categories: e.g. if the objects correspond to system components, and morphisms to links between

them (the interpretation of which depends, for example, on the type of architectural view being considered), a functor is a translation between two different architectural views.

Taking a step back, consider the category of which the objects are functors: the morphisms are then the *natural transformations*. The idea is not to transform a category into another category by observing how the subjacent relational categories interact (this was the role of functors), but to see how this transformation may be parametrized at the level of the objects which make up the category. Thus, we model the situation where a global transformation between complex architectures is carried out through local transformations at different levels of zoom.

Given two functors, there is an *adjunction* between them when there is a natural transformation between the identity and their composition; this provides a weaker notion than the existence of an inverse and is actually one of the main concepts of category theory, since it provides a general way to relate different notions.

We shall now introduce one last notion, that of *limit*. Let us take a diagram (i.e. objects and arrows between them): an object is "limit" if there are morphisms from this object towards each object in the diagram and if, in addition, each object satisfying this same property is such that there is a single morphism between it and the limit object. Let us consider how this principle can be applied to the modeling of complex systems: take the basic elements of a complex system as objects, and the relationships between them (energy flow, channels of transmission, proximity relationships, transition functions etc.) as arrows, producing a diagram within some adequate category. A limit object represents a sort of invariant associated with the diagram, which may then be seen as its internal organization. If we apply this kind of approach recursively, we can define a hierarchy where, at each level, the objects are limits of objects at the lower level. A representation of this kind is coherent with the idea that complexity is a relative notion and depends on the level of observation. Applied to different views of the systems which make up a system of systems, for example, it also provides a means of describing the links and hierarchies between these different visions.

Once you have a category, you may add further structure and obtain various types of categories. For instance, one can add an internal operation, usually denoted by $\otimes$, and ask for that operation to be associative up to a natural isomorphism (i.e. a natural equivalence which is an isomorphism), to have a right (resp. left) neutral element also up to a natural isomorphism: you get then a *monoidal category*. If the operation is also commutative up to a natural isomorphism, you get a *symmetrical monoidal category*. Furthermore if each object has a dual (a weaker notion than an inverse up to natural isomorphism), you get a *compact closed category*. If the operation $\otimes$ is a Cartesian product (the Cartesian product of two elements can be defined as the limit of the diagram composed of both objects), you get a *Cartesian category*, and if the product has an adjoint, you get a *Cartesian closed category* (this allows defining the set of all morphisms from one object to another as an object of the category, and is one of the leitmotifs of functional programming: functions are data). If there is a power object (a notion similar to the power set we are familiar with), then you get a *topos*.

Logics. We assume the reader is familiar with classical logic with its operators ∧ (conjunction), ∨ (disjunction) and ¬ (negation), and possibly with intuitionistic logic (which does not accept the classical axiom "tertium non datur" p∨¬p, hence there is a constructive flavor to the logic). We will discuss here another logic used increasingly in computer science (e.g. programming languages, parallel computing): linear logic [13]. The latter has two pairs of conjunction and disjunction-like operators related by duality (denoted by $^{\perp}$) that plays the role of negation (⊗ and ⅋ which are called multiplicative operators, & and ⊕ which are called additive operators) and a pair of dual quantifiers (! and ?) which model the consumptions of resources. Indeed in linear logic, like in a chemical reaction, if A is used to prove B, then A cannot be used again, unless you use !A which means that A can be used again and again, as is the case in classical or intuitionistic logic.

In order to get an idea at what the various operators mean, let us take the "menu example", similar to what Jean-Yves Girard gave when he introduced linear logic (remark: in the following formula, A⊸B stands for $A^{\perp}$⅋B): price⊸(fish&meat)⊗(cheese&(orange⊕apple)). For a given amount of money (which you lose once you order), you get a main dish and a second course; as main dish, you may choose between fish or meat; as second course, you may choose between cheese or seasonal fruit; however you do not choose the seasonal fruit, this is done by the restaurant.

Back to system theory, ⊗ models sequential composition (the notation is overloaded by tradition: do not confound with the previous tensor product), & parallel composition where the alternatives are built in the system (an "internal" alternative), ⊕ on the other hand is a parallel composition where the alternatives are driven by events external to the system (an "external" alternative), and ⅋ is a sequential composition that consumes all the resources in the prequel (the sequel is exchanged for the prequel: it can be seen as a sequential composition with consumption).

Linear logic, with its accounting of consumed resources, permits representing the step-by-step behavior of various abstract machines or processes: it is possible to describe the instantaneous state of the system and its stepwise evolution (each derivation of a new theorem consumes one time step and the state is transformed irreversibly into its successor) in an intrinsic way within the logic itself, i.e. without any explicit time parameters. Besides, using the analog of conjunction, disjunction and negation, the various segments of linear logic provide a way to express requirements, and the subtleties of the operators dealing with internal and external choice can model both design-driven and event-driven alternatives. If input or output sequences are available, logical formulae can be written, logical properties can be defined on system behaviors, and the latter can be checked for logical properties.

In applications, linear logic can be used restricted either as multiplicative linear logic (only multiplicative operators are used), or multiplicative additive linear logic, with classical or intuitionistic variants depending on whether a constructive flavor is needed or not, or full linear logic with all operators.

As a matter of fact, well-known models of processes or systems such as Petri nets have been shown to relate to linear logic. Recall that Petri nets are models in which (instances of) places (i.e. tokens) can be understood as available resources, and transitions as concurrent activities that require exclusive use of some of these resources and that, after completion, release new resources (tokens in places) to the environment. Pre- and post-conditions have to be satisfied to trigger a transition. This model has been extended in various directions, leading to colored nets, higher-level nets, etc. In [10] it is shown how Petri nets form models of intuitionistic linear logic, in [5,6] correspondences are built up between Petri nets and certain formulae of fragments of linear logic. E.g. the formula $!(A \otimes B \otimes B \multimap D \otimes E)$ encodes a Petri net transition that can be persistently fired and needs one taken of place A and two of place B, and adds one token to place D and one to place E.

Categorical Logic. This notion provides the link between categories and logics. It relies on the fact that for any logic it is possible to construct an ad hoc category that is a model of the given logic (this approach is called *categorical semantics*): the general idea is that an object will correspond to each formula and morphisms correspond to equivalence classes of proofs (there is a morphism between two objects if there is a proof between the corresponding formulae). Conversely, to any category it is possible to assign a logic (this approach is known as the *internal language* approach): the general idea is to assign types to objects, and typed terms to morphisms; this builds the so-called internal language of the category which has an intrinsic logic, depending on the properties of the initial category.

Both previous processes are not inverse, but rather adjoint (starting from a category, you get a logic, from which you get another category that is not the initial one, but can be compared to it through a natural isomorphism). This defines natural equivalences between classes of categories and classes of logics. Various types of categories correspond to various logics, which helps zigzagging from one theoretical framework to the other. For instance, untyped lambda-calculus corresponds to Cartesian closed categories, intuitionistic logic corresponds to toposes, multiplicative additive linear logic corresponds to compact closed categories.

3 Formal Introduction of the Framework

3.1 A Category of Systems

Several viewpoints are admissible with input-output systems: you consider either a system with inputs coming in and outputs coming out, or you focus on a link (flow of matter, energy or information) connecting two systems. Both representations are dual and depend on the community dealing with system issues: control theorists or systems engineers use the first system-oriented viewpoint, while computer scientists would prefer the second process-oriented viewpoint, by focusing on the transformation.

The framework presented below deals obviously with both viewpoints. However, since we do systems engineering, we use the first interpretation to illustrate the concepts. We will deal with systems with m inputs and n outputs, which will be graphically interpreted as boxes and wires. We introduce the *category Σ of such systems*: objects label the wires which enter and exit a box, morphisms label the boxes, and a composition operator $(Y \to Z) \circ (X \to Y)$ allows sequential composition. As a convention we define a standard orientation (top down in the figures), which means that unless precised, all wires are oriented one way (see it as the flow of time). Input/output interfaces are controlled by grouping wires together and composing systems. To do that, we introduce a tensor product $X \otimes Y$ that allows parallel composition and make *Σ a symmetric monoidal category*.

Recall that a monoidal category is a category C, with a bifunctor $\otimes: C \times C \to C$, a unit object I of C, and natural isomorphisms $a_{X,Y,Z}: (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$, $l_X: I \otimes X \to X$, $r_X: X \otimes I \to X$, subject to the well-known coherence axioms. When there is in addition a natural isomorphism $c_{X,Y}: X \otimes Y \to Y \otimes X$, C is symmetric monoidal.

Graphically, arrows from a tensor of m objects to a tensor of n objects in a monoidal category may be represented by boxes with m inputs and n outputs. The symmetry is a twist of the wires and allows arrows to overcross.

Such a choice of a base category reflects our visual intuition about flow graphs. However it is too restrictive to deal with all possibilities offered by system description languages, such as used in SysML (when considering ports and channels). Interaction graphs [1] are more adequate: instead of considering only one-way arrows, we accept two-way communication for every arrow. This allows modeling complex interfaces: there is a client interface (with A^+ coming in and A^- coming out) before the system box, and a server interface (with B^+ coming out and B^- coming in) after the system box. Notice that such an interaction graph can be formally written as a flow graph by repositioning the arrows (see Fig.2).

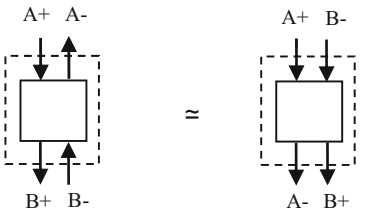


Fig. 2 Translation of an interaction graph to a flow graph

Mathematicall, this will lead us (see below) to consider compact closed categories (duals are needed to change upper-oriented arrows into down-oriented arrows, with our orientation convention), instead of symmetric monoidal categories.

Interaction graphs can be composed sequentially, and translating this operation to flow graphs illustrates the necessity of mathematical operators for “crossing-over” and “back-tracing” wires.

This introduces *feedback*, when the output wire makes a loop and reenters the system as an input. The name comes from control theory; however in computer science, one faces also the situation where two processes feed each other some data, and process them together, i.e. partially evaluate over them; both processes thus interact when a part of the outputs of one of them is piped to the input interface of the other, and vice versa (like in Fig. 3). This interaction is often seen as iteration and the result as a fixed point in some form of abstract calculus. Anyway, it is a key ingredient of advanced system theory, whatever the application domain.

In order to define correctly feedback operations in a categorical theory setting, it is necessary to have an intrinsic description. If the definition relies on particular properties of the objects of the category (such as [14, 15]), it is ad hoc and does not take advantage of the categorical setting: proofs do not involve then categorical arguments and lack the universality that theory may offer. It is also awkward to establish a characterization of the categorical definition in terms of the logical interpretation, which weakens the formal foundation.

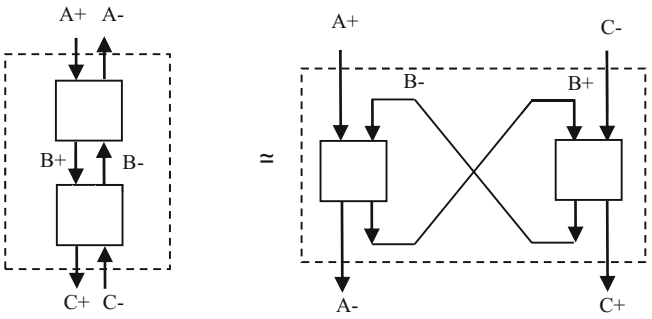


Fig. 3 Sequential composition of interaction graphs; translation to a flow graph composition

The formal definition is [8, 23, 35]: a trace for a symmetric monoidal category is a natural family of functions $Tr^{U}_{A,B}: C(U \otimes A, U \otimes B) \rightarrow C(A,B)$, called trace over U of a morphism $A \rightarrow B$, satisfying axioms known as vanishing (tracing over the unit I has no effect ; tracing over U and then over V is the same as tracing over $U \otimes V$), superposing (tracing over U and tensoring the result with g is the same as tracing $f \otimes g$ over U), yanking (tracing $c_{U,U}$ over U yields identity), sliding (tracing $(g \otimes 1_A) \circ f$ over U is the same tracing as $f \circ (g \otimes 1_B)$ over V). A category is traced when a trace exists for all U.

Graphically, a trace joins the output wire U to the input wire U, and is the archetype of the feedback loop.

Actually, we will not make Σ a traced symmetric monoidal category, but go one small step further and make Σ *a compact closed category*.

Recall that a symmetric monoidal category is compact closed if any object A has a left dual A^* with morphisms $\varepsilon_A: A^* \otimes A \rightarrow I$ (the counit) and $\eta_A: I \rightarrow A \otimes A^*$ (the unit) satisfying additional so-called yanking conditions.

Graphically, A^* labels the arrow pointing in the opposite direction of the arrow labeled by A. η_A corresponds to a wire coming out of the box and returning into it, while ε_A does the same but at the input interface.

The relationship between traced monoidal categories and compact closed categories is a strong one: every compact closed category admits a so-called canonical trace (defined uniquely by using the appropriate units and counits over U and U^*). Conversely, given a traced monoidal category C, there is a fully faithful embedding of it into a compact closed category *Int(C)*. The objects of *Int(C)* are pairs *(A+, A-)* of objects of C, a morphism of *Int(C)* from *(A+, A-)* to *(B+, B-)* is given by a morphism of the form $A+ \otimes B-$ to $A- \otimes B+$ in C, and composition of two such morphisms is given by tracing out $B+$ and $B-$.

These structures may seem unfamiliar to some readers, but they have been used in the last decades for instance to better understand the algebraic structures of Petri nets [37,41].

Up to now we have given Σ the minimal categorical structure necessary to formalize usual activities done with widely used system description languages. It is worth noticing that similar ideas have been raised recently in control theory [3,4].

3.2 Closing the Loop with Categorical Logic and Proof Theory

Once we have this categorical framework, we can now delve into categorical logic: the keystone of our general approach is that linear logic (actually !-free multiplicative intuitionistic) has categorical semantics in compact closed categories. Additive multiplicative intuitionistic linear logic corresponds to compact closed categories with products, and full intuitionistic linear logic corresponds to linear categories (compact closed categories with a comonad satisfying several properties ensuring compatibility between both structures). It should be emphasized that the relationship between classes of logics and categories goes beyond a one-way inclusion relationship: there is actually an equivalence between the corresponding classes (seen as categories), based on the result that each logic calculus provides an internal language for the corresponding class of categories.

Therefore it is possible to model a system as a category, and specify properties about the latter within the relevant linear logic fragment. Going from the considered classes of categories to the relevant classes of logics can actually be interpreted as a functor: it is in fact the construction of the internal logic from suitable structured categories to logical theories. This functor has a left adjoint functor, known as the construction of the syntactic category of a theory. The interesting fact is that this exhibits an adjunction between logics and categories.

As linear logic has a well-studied proof theory (e.g. see proof nets, and geometry of interaction), which furthermore relies on graphic interpretations that are none else than string diagrams like those we illustrated before, the modeling framework provides simultaneously the verification framework.

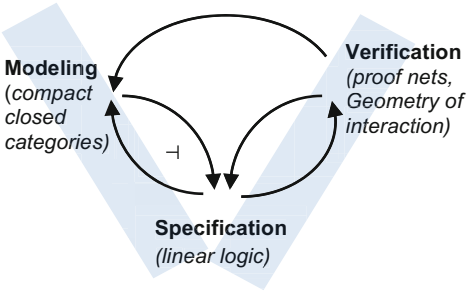


Fig. 4 The formal foundation of systems engineering, with its underlying adjunction

In summary we have the formal framework depicted in Fig.4, which relates together the three main steps to the systems engineering lifecycle process, and can also be seen as a recursive formalization of the waterfall process.

Since linear logic encompasses other logics, like classical logic or intuitionistic logic, and since the corresponding categorical semantics are folklore (for instance, first order intuitionistic models are toposes, which are special symmetric monoidal closed categories [40], where the tensor product is the Cartesian product, $a_{X,Y,Z}$ is an isomorphism, l_X and r_X are the projections given by the product), the same modeling/specification/verification activities can be done for those logics too. *That shows the general flexibility of the proposed framework for various usual languages of different power of expressiveness.*

3.3 Extensions of the Proposed Framework

The previous paragraphs only give a broad overview of the use of category theory in engineering (modeling, specifying, verifying) complex systems. To go further, we could enrich objects in terms of structure, particularly to take aspects of dynamic evolution into account [26, 27, 28].

For instance, objects can be endowed a presheaf structure: e.g. each object can be seen as a functor from the category of sets into a given category. Actually, instead of the category of sets, it is possible to consider another structure such as subsets of reals, or a group or a semi-group (what is needed at least is a partial order on that structure in order to define a presheaf). This is then nothing else than a time-varying state, where various time structures can be modeled easily: continuous (the group would be the real line or the semi-group the positive reals), discrete (the semi-group could be the integers), or hybrid (subsets of the nonstandard reals can be considered, or semi-groups such as in [15]). The image category provides additional algebraic structure on the state (take an Abelian category for instance, and you can do some linear theory on the states). Using the general framework described earlier with such objects, we obtain a categorical framework of control theory where the states are presheaves, and state-feedback controlled systems fit naturally as the objects of a compact closed category. By taking

advantage of the correspondence between various categories and logics, logical properties of control theory such as liveness, deadlock, or reachability issue can be shown.

Enriching the base category with additional structure, for instance with finite products (Cartesian category, for either the tensor product or another operation) or more generally any limits (complete category) is another possibility. However, if we assume the existence of products, we have to be careful, since this yields for instance $X \times I \simeq X$, which means for a system that duplication followed by deletion of one of the copies is as if nothing had happened. This might be fine for a Web service but will obviously not always be the case in the real world. More specifically, a symmetric monoidal category is a Cartesian category (i.e. $\otimes$ is a product) if and only if there exist monoidal natural transformations (natural transformations compatible with the tensor product) $d_A : A \to A \otimes A$ and $e_A : A \to I$ defining a comonoid for every object A. One recognizes the duplication and deletion operators illustrated in Fig. 1. Therefore special attention should be paid to the interaction between product and tensor operations.

However, on the other hand, existence of products or more generally of limits has also its advantages, since from several systems with different features, it is then possible to construct a system that subsumes them (the colimit) or that has only the common features shared by the system (the limit). To go further into that direction, one could notice that the category of systems we introduced previously can be related to the category *Span(C)* of spans [25, 28] built on a given category C: its objects are diagrams $X \leftarrow Y \to Z$ (modeling for instance a service Y putting into relation a producer Z and a consumer X) where X, Y and Z are objects in C. This category can be shown to be complete if C has pullbacks, and is compact closed if C is complete.

The best is thus to define two types of morphism composition: one which yields a Cartesian category and another which yields a symmetric monoidal category. In this case, there is no collapse of the various structures, and we have the advantages of the various formal models. However the link with the categorical logic side is more complicated, since we shall not have a nice adequation like previously: either we have to introduce an intermediate forgetful functor and work independently on either facet of the categorical model, or we have to introduce a more complex logic and walk into less well-trodden ground!

Another extension not developed in this paper is the use of monads (a functor equipped with natural transformations that equip it with a monoid-like structure) in addition to the compact closed category. On the one hand monads have been seen as models of computations [33] and arise also naturally in coalgebraic approaches ([22] shows how coalgebraic traces, in suitable Kleisli categories, give rise to traced monoidal structure in those Kleisli categories, with finite coproducts as monoidal structure). On the other hand, compact closed categories with monads (more precisely the comonad arising from a pair of adjoint functors of such a category to a Cartesian closed category) define classes of categories such as Seely categories, Lafont categories and linear categories, that correspond to linear logic (the monads are necessary to take the operator ! into account).

This is another a posteriori confirmation that the proposed framework seems adequate, since it encompasses naturally (i.e. by remaining in the spirit of the categorical internal approach) many existing formalisms, and it keeps the step-by-step coherence between the categorical and the logical approaches.

4 Concluding Remarks

In this paper we have presented a formal approach of system engineering, where the model-specify-verify process is formalized by adjunctions between specific categories, logics and proof theories. This relies on the use of symmetric monoidal and compact closed categories, which appear as well adapted to current modeling tools and languages.

Although over the last decades, monoidal categories, and more precisely compact closed and traced monoidal categories, have received some attention from theoretical computer science, especially in concurrency theory [38], such work does not seem to have been applied outside computational systems, except very recently for quantum systems (due to the recent advances in quantum computing, especially cryptography). We see a real advantage of the application of such a theoretical framework to systems engineering, as it provides a unified basis to the key steps of systems engineering and uses furthermore concepts directly related with wide spread modeling practices such as graphical description languages.

References

1. Abramsky, S., Gay, S., Nagarajan, R.: Interaction categories and the foundations of typed concurrent programming. In: Proceedings of the 1994 Marktoberdorf Summer School. NATO ASI Series. Springer (1995)
2. Alexiev, V.: Applications of linear logic to computation: an overview. TR93-18, Univ. of Alberta, Canada (1993)
3. Baez, J.: Categories in control. Talk at Erlangen University. Downloadable as johncarlosbaez.wordpress.com/2014/02/06/categories-in-control (2014)
4. Baez, J., Erbele, J.: Categories in control. Downloadable as arXiv.1405.6881v1[math.CT] (2014)
5. Brown, C.: Relating Petri nets to formulae of linear logic. LFCS Report Series ECS-LFCS-89-87, Univ. of Edinburgh (1989)
6. Brown, C., Gurr, D., de Paiva, V.: A linear spcification language for Petri nets. TR-DAIMI-PB-363, Univ. of Aarhus (1991)
7. Cattani, G.L., Winskel, G.: Presheaf models for concurrency. In: van Dalen, D., Bezem, M. (eds.) CSL 1996. LNCS, vol. 1258, pp. 58–75. Springer, Heidelberg (1997)
8. Bagnol, M., Guatto, A.: Synchronous machines: a traced category. Draft downloadable as hal.inria.fr/hal-00748010 (2012)
9. Coecke, B., Paquette, E.O.: Categories for the practising physicist. Downloadable as arXiv.0905.3010v2[quant-ph] (2009)
10. Engberg, U.H., Winskel, G.: Linear logic on Petri nets. BRICS Report Series RS-94-3, Univ. of Aarhus (1994)

11. Fiadeiro, J.L.: Categories for Software Engineering. Springer, Berlin (2005)
12. Fiori, C.: A first course in topos quantum theory. Lecture Notes in Physics, vol. 868. Springer (2013)
13. Girard, J.-Y.: Linear logic. Theoretical Computer Science, London Mathematical 50(1), 1–102 (1987)
14. Golden, B., Aiguier, M., Krob, D.: Complex systems modeling II: a minimalist and unified semantics for heterogeneous integrated systems. Applied Mathematics and Computation 218(16), 8039–8055 (2012)
15. Golden, B.: A unified formalism for complex systems architecture. Ph.D. in Computer Science, Ecole Polytechnique (2013)
16. Goguen, J.: Categorical foundations for systems theory. In: Pichler, F., Trappl, R. (eds.) Advances in Cybernetics and Systems Research, pp. 121–130. Transcripta Books (1973)
17. Goguen, J.: A categorical manifesto. Mathematical Structures in Computer Science 1(1), 49–67 (1991)
18. Goguen, J.: Sheaf semantics for concurrent interacting objects. Mathematical Structures in Computer Science 11 (1992)
19. Grosu R., Stauner T.: Modular and visual specifications of hybrid systems: an introduction to HyCharts. TUM-I9801, Technische Universität München (1998)
20. Grosu, R., Broy, M., Selic, B., Stefanescu, G.: What is behind UML-RT? In: Behavioral Specifications of Businesses and Systems, pp. 73–88 (1999)
21. Heunen, C., Sadrzadeh, M., Grefenstetter, E.: Quantum physics and linguistics. Oxford University Press (2013)
22. Jacobs, B.: From coalgebraic to monoidal traces. Electric Notes in Theoretical Computer Science Proceedings of Coalgebraic Methods in Computer Science (2010)
23. Joyal, A., Street, R., Verity, D.: Traced monoidal categories. Math. Proc. Camb. Phil. Soc., vol. 119, pp. 447–468 (1996)
24. Katis, P., Sabadini, N., Walters, R.F.C.: Bicategories of processes. Journal of Pure and Applied Algebra 115, 141–178 (1997)
25. Katis P., Sabadini N., Walters R.F.C.: On the algebra of feedback and systems with boundary. Rendiconti del Circolo Matematico di Palermo, Serie II (suppl. 63) (1999)
26. Luzeaux, D.: Towards the engineering of complex systems. Journées Nîmes 98 sur les Systèmes complexes, systèmes intelligents et interfaces, Nîmes, France (1998)
27. Luzeaux, D.: Category theory applied to digital systems theory. In: 4th World Multiconference on Systemics, Cybernetics, and Informatics, Orlando, FL, USA (1998)
28. Luzeaux, D.: Vers une nouvelle théorie abstraite des systèmes en vue de leur ingénierie. In: 5e conférence annuelle d'ingénierie système de l'AFIS, Paris (2009)
29. Mac Lane, S.: Categories for the working mathematician, 1st edn. Springer, New York (1971)
30. Mac Lane, S.: Categories for the working mathematician, 2nd edn. Springer, New York (1998)
31. Mac Lane, S., Moerdijk, I.: Sheaves in geometry and logic: a first introduction to topos theory. Springer (1968)
32. Malherbe, O., Scott, P., Selinger, P.: Presheaf models of quantum computation: an outline. Downloadable as arXiv:1302.5652v1 (2013)
33. Moggi, E.: Notions of computations and monads. Information and Computation 93(1), 55–92 (1991)

34. Nielsen, M., Sassone, V., Winskel, G.: Relationships between models of concurrency. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1993. LNCS, vol. 803, pp. 425–476. Springer, Heidelberg (1994)
35. Pavlovic, D.: Tracing the man in the middle in monoidal categories. Downloadable as arXiv:1203.6324v1 (2012)
36. Rao, J., Küngas, P., Matskin, M.: Composition of semantic Web services using linear logic therorem proving. Information Systems 31, 340–360 (2006)
37. Sassone, V.: An axiomatization of the category of Petri net computations. Mathematical Structures in Computer Science 8, 117–151 (1998)
38. Selinger, P.: Categorical structure of asynchrony. Electric Notes in Theoretical Computer Science 20 (1999)
39. Selinger, P.: A survey of graphical languages for monoidal categories. Downloadable as arXiv:0908.3347v1 (2009)
40. Slodicak, V.: Toposes are symmetric monoidal closed categories. Scientific Research of the Institute of Mathematics and Computer Science 1(11), 107–116 (2012)
41. Winskel, G.: Petri net algebras, morphisms and compositionality. Information and Computation, 197–238 (1987)

Ontology-Assisted Systems Engineering Process with Focus in the Requirements Engineering Process

Anabel Fraga, Juan Llorens, Luis Alonso, and José M. Fuentes

Abstract. Problems found in the current Systems Engineering with focus in the Requirements Engineering Process shown that it could be improved using ontologies for aiding in the process. Requirements engineering in the Systems Engineering process is enhanced and quality of requirements enriched as well, improving Systems Engineering capabilities clearly can result in better Project Performance. One of that is the Requirement improvement and of course the benefit goes to the whole process of development. The more correct, complete and consistent it is, the best performance it will have and ontologies enable a more exhaustive and fast quality process.

1 Introduction

The Systems Engineering Division (SED) of the National Defense Industrial Association (NDIA) established the Systems Engineering Effectiveness Committee (SEEC) to obtain quantitative evidence of the effect of Systems Engineering (SE) best practices on Project Performance. The SEEC developed and executed a survey of contractors for the defense industry (i.e., government suppliers) to identify the SE best practices utilized on defense projects, collect performance data on these projects, and search for relationships between the application of these SE best practices and Project Performance. As shown in the

Anabel Fraga · Juan Llorens · Luis Alonso
Universitad Carlos III de Madrid, Madrid, Spain
e-mail: {afraga,llorens,luis.alonso}@inf.uc3m.es

José M. Fuentes
The Reuse Company, Madrid, Spain
e-mail: jose.fuentes@reusecompany.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management,*
DOI: 10.1007/978-3-319-11617-4_11

report [16], improving Systems Engineering capabilities clearly can result in better Project Performance. One of that is the Requirement improvement and of course the benefit goes to the whole process of development. The more correct, complete and consistent it is the best performance it will have.

The most common defects encountered within requirements were the ambiguity and expressing needs in the form of solution [14]. The adequate requirements management is the most important factor in the success of any Project, even more tan tests, design and programming. If you don't know what you want, you don't know where you go.

The application of ontology engineering in systems engineering seems to be a very promising trend [20], [7]. We call it system verification based on Knowledge Management, and it deals with assisting System Engineers to get a complete and consistent set of requirements (e.g. compliance to regulation, business rules, non-redundancy of requirements…) by using Ontologies, which represent the domains of knowledge of an organization. The combination of Requirements Engineering with Knowledge Management, throughout Information Retrieval from existing sources, allows the verification process to measure quality of a set of requirements by traceability, consistency/redundancy, completeness and noise. Information retrieval enables also to verify the completeness of the ontology using a PDCA (Plan-Do-Check-Act) cycle of improvement. Requirements engineering is the first step and by traceability and Ontology based systems, similar assets of any phase of the development process used in analogous projects could be reused and adapted to a new challenge.

For instance, by using a semantic approach, a requirement can be translated into a graph and by means of NLP (Natural Language Processing) techniques. It could be compared with another requirement or test or document by similarity, as for instance:

UR044: The Radar shall be able to detect hits at a minimum rate of 10 units per second

This example will be used to illustrate the rest of the sections.

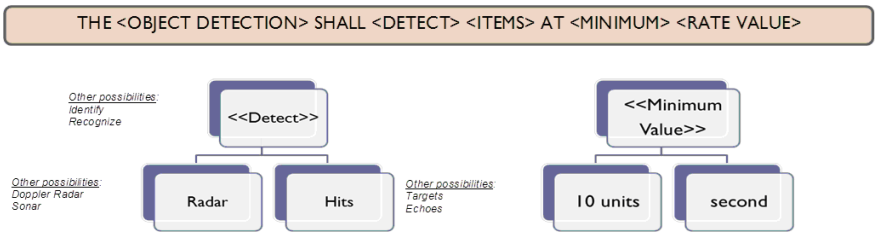


Fig. 1 A requirement similarity comparison

The use of a semantic tool for representing the knowledge of the requirement (for instance) allows us to compare diverse requirements written in a different way because what matters is in this case the minimum value of detection and it must be 10 seconds, if another requirement says it must be 15 seconds, then a conflict or contradictory requirements for the radar are available in the Systems requirements which means a huge error in calculus if it is an important measure for the radar. Extending NLP with inference rules capabilities, taxonomies, thesaurus and semantic groups enable computer tools to enhance systems engineering requirements, models, architectures, tests or documentations consistency. Even though, 10 seconds could be compared in another range of metrics in order to guarantee that this value is unique within the requirements of the system.

One of the problems found is the similarity of requirements, incompleteness, different use of measurement units, and so on. It is hard to compare sequentially a group of thousands of requirements. Most common requirement defects are the following: not verifiable, not precise enough, several requirements gathered in a single one, lack of consistency, not completeness, ambiguous, requirements expressed as solutions, and so on. An ontology-assisted System Engineering Process can sort out these problems in a reasonable time.

The reminder of this paper is structured as follows: Section 2 explains how is nowadays the requirements engineering process, Section 3 explains how ontologies are considered and applied in the ontology-assisted requirements engineering process within the systems engineering process, Section 4 explains an overall application of the ontology-assisted vision in the systems engineering process, and finally Section 5 includes some conclusions.

2 How Is Nowadays the Requirements Engineering Process

A requirement is an identifiable element of a function specification that can be validated, and against which an implementation can be verified. [3] [10] [9] [19] [8] [6] [5] The main requirements attributes are mentioned below [8]:

- Each requirement must be uniquely identifiable.
- Verifiable: it must be very simple to determine the success/failure property for every requirement.
- Clear: it must describe what the software does, without involving other parts of the system.
- Practical: each requirement must be derived from a user need.
- Complete: describe the whole customers' situation case.
- Consistent: without internal conflicts between requirements.
- Correct: describes accurately and exactly the customer's situation and need.
- Modifiable: documented in a structured and accessible manner.
- Traceable: against whatever number of artifacts used in the life cycle.
- Accurate: the requirement should only be understood in only one way.

Software requirements engineering [3] [10] [9] [19] [8] is a disciplined, process-oriented to the definition, documentation, and maintenance of software requirements throughout the software development life cycle. Software Requirements Engineering is made up of two major processes: requirements development and requirements management. [7]

- Requirements Development encompasses all of the activities involved in eliciting, analyzing, specifying and validating the requirements.
- Requirements Management encompasses the activities involved in requesting changes to the requirements, performing impact analysis for the requested changes, approving or disapproving those changes, and implementing the approved changes. Furthermore, includes the activities used for ensuring that work products and project plans are kept consistent and tracking the status of the requirements as one progresses through the software development process.

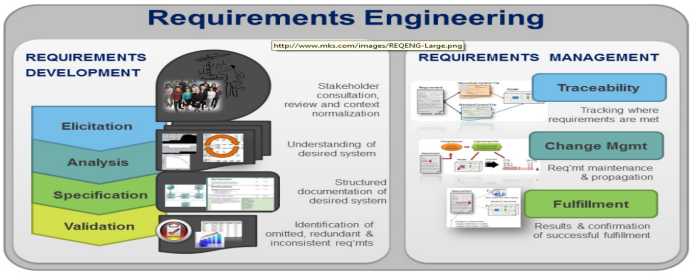


Fig. 2 Requirements Engineering Process by PTC Company. [17]

Requirements specification produces a formal software requirements document from the set of requirements. Its purpose is to give visibility of the requirements to the system/test engineers and to allow the formal verification of the requirements specification. [3] After that, requirements verification is the final process that ensures that the requirements are correct. It represents the formal agreement and acceptance of the requirements that the software must implement. It ensures that the specification is complete and that the requirements are feasible in terms of its technical implementation and verification, taking into account time/budget constraints. Once this process is finished, the software requirements document is formally issued and constitutes the technical baseline for further developments.

The requirements management process handles the modification of the software requirements after the document has been formally reviewed and agreed. Thus, the activity is done in order to ensure that the baseline of the software is known and to analyze affordability of change in terms of budget and design. The change is proposed by the system engineers and must be agreed by the software development team. Once agreed, the change is included in the software requirements document and a new baseline is established.

There are some rules that establish how the requirements must be written and which mistakes must be avoided. The INCOSE (International Council on Systems Engineering) rule states that the requirements must be clear and concise, complete, consistent, verifiable, affordable, achievable, necessary, bounded, traceable with independent implementation. [11]

The SMART (mnemonic criteria to guide in the setting of objectives) criteria define that a requirement must be Specific, Measureable, Achievable, Relevant and Traceable. It is important to keep the requirement as simple as possible, avoiding unnecessary information. [11]

A suggested list of requirement attributes is [8]:

- Write using the same words with exact meaning established.
- Utilize clear, unambiguous phraseology and punctuation.
- Do not misplace comas.
- Use correct verb tense: o Shall - a demand. o Will - a future happening. o Must - a strong desire. o To be, is to be, are to be, should, should be - nice to have, desired capabilities.

The use of certain words should be avoided, because they can convey uncertainty:

- Superlatives such as “best” and “most”.
- Subjective language: “user friendly”, “cost effective”, “Easy to use”…
- Vague pronouns: he, she, this …
- Ambiguous adverbs and adjectives: “minimal”, “almost always”, “significant”, “quick/rapid/safe”, “sufficient”,…
- Open ended, non-verifiable terms: “provide support”, “but not limited to”,…
- Comparative phrases: “better than“, “higher quality”
- Loopholes: “as applicable“, “if possible”
- Other indefinites: and so on, TBD, TBC, etc.

Other important attributes of good requirements are:

- Identify all stakeholders across all products lifecycle phases and involve them during requirements development and validation in order to build what is really needed.
- Take ownership of requirements.
- Always use the imperative shall and identify subject, result and success criteria in measurable terms.
- Requirements shall be uniquely and identified.
- Write requirements clearly and explicit.
- Requirements must be verifiable in order to allow proving that they have been met.
- Justify each requirement by a rationale and/or trace to its source.
- Allocate and flow down requirements.

Therefore it is necessary to comply with these rules avoiding possible risk of failure in the implementation and in the final product that results in an unsatisfied customer or corporative damage.

Any mistake in the requirements definition phase is distributed downwards until low level requirements being almost impossible to fix. Thereby, those mistakes must be caught in the early development process.

When defining any system we will have a set of requirements that attends to their relation and dependences among them, which have to comply with the CCC philosophy: Consistency, Completeness and Correctness. [18] Completeness means that the set of requirements does not need further improvement because it contains everything concerning the system definition. Consistency states that the requirements do not have contradictions within them, not duplicated or even that a term is used for the same semantic in all the requirements.

3 Ontology-Assisted Requirements Engineering Process in the Systems Engineering Process

The solution developed to improve the problems when writing requirements is establishing ontological structures.

It can be defined as a common vocabulary in which shared knowledge is represented. A specification of a representational vocabulary for a shared domain of discourse (definition of classes, relations, functions and other objects) is called ontology. It is an explicit and shared specification of conceptualization. Ontology is a knowledge-base within the system development process, which has information about the structure of the System, in the subject of application of the written requirements. It consists of controlled vocabulary, thesaurus, light ontology and full-ontology with patterns and representation schemas. [14] In a nut shell, this is the vision of the ontology:

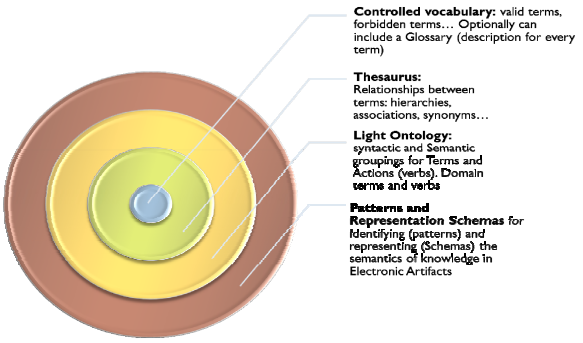


Fig. 3 Ontology Layers [9]

3.1 Controlled Vocabulary

It is needed for standardizing and normalizing the terminology used in the custom application. The input information must/should match the controlled vocabulary. Using a glossary with different categories of terms, the ontology may store:

- Client business related Terms: those terms focused into the customer area to be considered.
- General Language Terms: those terms related to the idiomatic field.
- Syntactically relevant phrases: Adverbs, Adjectives, and so on.
- Invalid terms: those terms that could be of no relevance.

Following the example introduced in the first section, the terms extracted from that requirements are shown in Fig. 4.

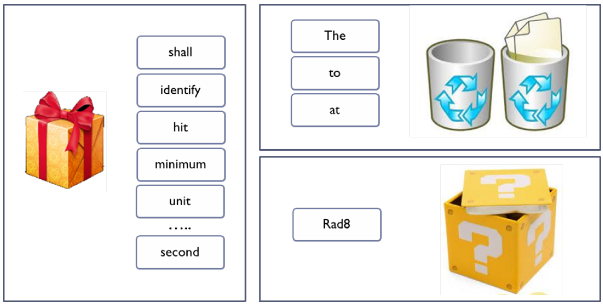


Fig. 4 Stop words, general language terms and business terms

3.2 Thesaurus

A Thesaurus stores relational information regarding the terms in the glossary. It is used for:

- Retrieval purposes
- Representation and normalization purposes
- Suggestion purposes (Decision support)
- "solution specific" purposes

It enriches the controlled vocabulary for including specific relationships of the represented domain: synonyms, hierarchies, and general associations. Following the example, a new requirement is introduced:

UR03442 : The Radar shall be able to distinguish hits at a minimum rate of 10 elements per s

Fig. 5 shows a relationship between Rad8 PTT and Rad8 of equivalence, as well between distinguish and identify, and between s and second. On the other hand,

Radar is a super class of Rad8, and due to the synonymy of Rad8 PTT. Also a generic relationship between Radar and Sensor is shown.

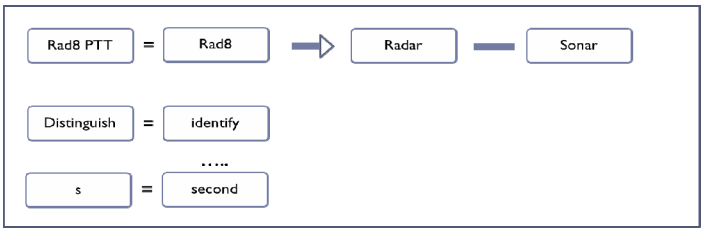


Fig. 5 Relationships in the requirements examples UR044 and UR03442

3.3 Light Ontology

The Light ontology contains:

- Syntactic Information: For NLP purposes and for specific Pattern Restrictions

An example of the kind of syntactic information to be represented following the example shown before:

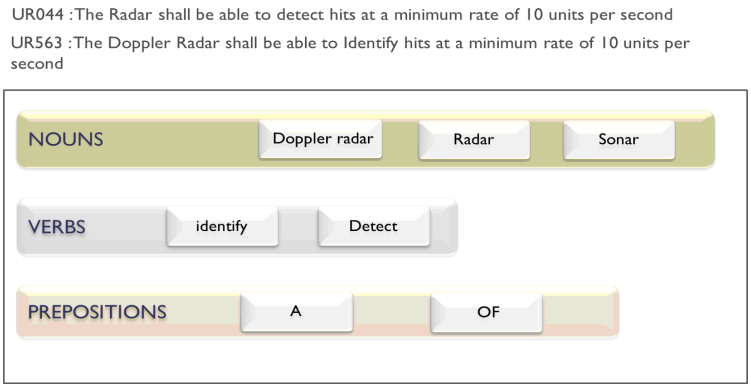


Fig. 6 Nouns, Verbs and prepositions recognized in the requirements

- Semantic Information: For Retrieval purposes and for specific Pattern Restrictions. The semantic information extracted from the requirements is shown as follows (Fig. 7).
- Idiomatic Information: For NLP purposes.
- Artifact Type Information: For Retrieval Filtering purposes.

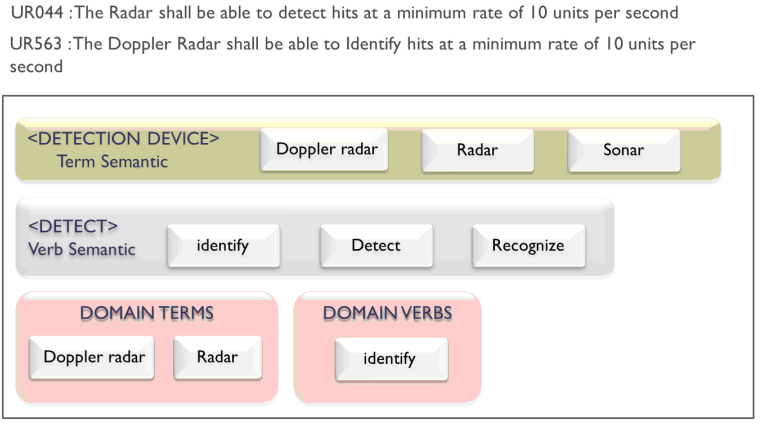


Fig. 7 Semantic grouping of verbs and terms, domain terms and domain verbs

3.4 Patterns and Representation Schemas

The patterns [12], also called boilerplates, are sequential restrictions based on a place-holders structure for the specific terms and values that constitute a particular knowledge statement, where the restrictions can be grammatical, semantic, or even both, as well as other patterns.

A pattern encapsulates the rules for writing and validating a knowledge statement of a particular kind. It may be possible to abstract away from the textual representation by using conceptual graphs in the style of [13]. It needs to be completed following these phases:

- Creating the detection pattern.
- Creating the formal representation of the knowledge statements based on the pattern information.

An example of a pattern created for the requirement example UR044 is shown as follows in Fig. 8.

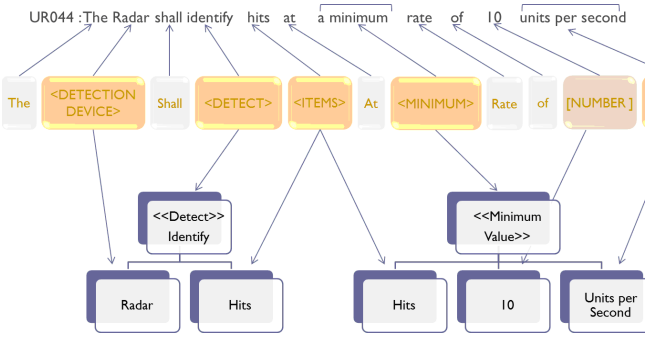


Fig. 8 A pattern and its knowledge representation

3.5 The Ontology in the Center of the Process

The appropriate selection of the knowledge structure allows different possibilities to the organization.

The System Knowledge Repository (SKR) allows representing, storing, managing and retrieving: Relevant knowledge around the System and its domain (including the SE Process), and also Digital content (Assets) regarding a particular System

The SKR is formed by:

- SKB – System Knowledge Base
- SAS – System Assets Store

The System Knowledge Base (SKB) supports the complete system knowledge-base for the application of semantic services around the system life cycle (Including SE). The System Assets Store (SAS) manages a formal representation of the System Assets: Requirements, Models, and so on. It is the base for offering services around these assets: Reuse, Traceability, MDE, TDD, etc.

3.6 Ontology Tools Available

Diverse tools are available for building ontologies [1], the most known among practitioners is Protégé [15], but it is difficult to build an ontology in any of the available tools and then use it for analyzing the meaning of the information available in the company. There is a suite of tools called Requirements Quality Suite (RQS) that contains a knowledge management system called knowledgeMANAGER [20], it is connected to the whole suite and a semantic use of the text is done when dealing with requirements within the system engineering process. The process and knowledge-assisted process is supported by this suite of tools [20].

4 Applied In

The application of ontologies in the systems engineering process and mainly in the systems requirements can be applied in:

4.1 Authoring

The image below shows the tool Requirement Authoring Tool (RAT). RAT is part of the Requirements Quality Suite (RQS) and leads authors during the process or requirements writing. The list box in the center of the screen represents the proper

grammar of the requirements, while the editing box (in the top of the screen) provides dropdown lists with the set of valid concepts for every part of the requirement.

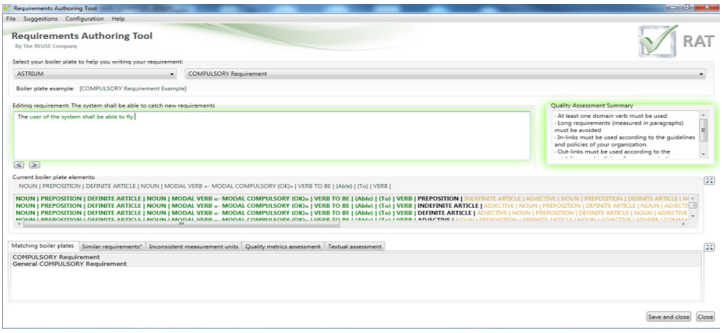


Fig. 9 Requirement Authoring Tool based on Patterns (Ontology).

It is important to note that, aside of providing intellisense support for writing the requirements, RAT also provide quality information on the fly according to the agreed set of metrics. The image below represents this quality analysis on the fly (Fig. 10).

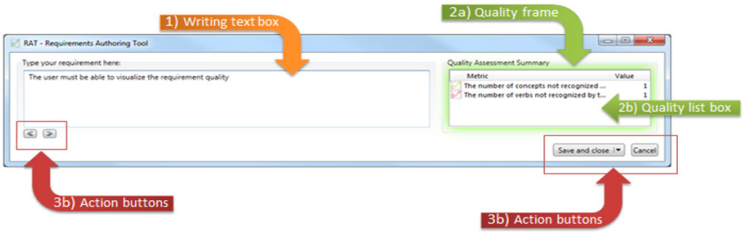


Fig. 10 Intellisense information detailed

4.2 Quality

The ontology will aid in the process of checking the CCC criteria, as shown in Fig. 11, two requirements that contains the same information but using different words are detected as duplicated requirements even they are not typed in the exact way, but the domain relationships in the ontology and also the semantic grouping aids in the detection of this kind of requirement, we can call it a quality detection system.

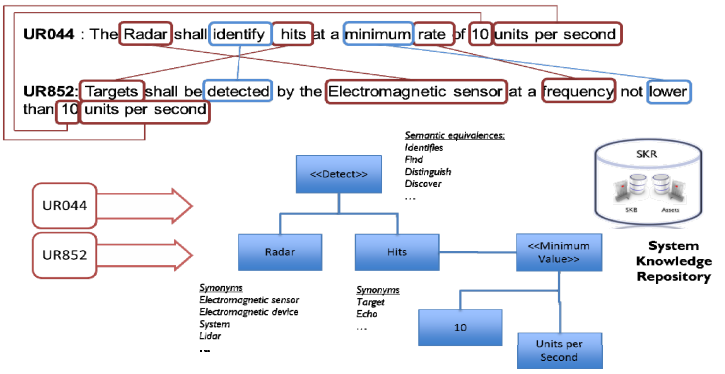


Fig. 11 Example of an ontology used for detecting duplicated requirement

There is one (similarity) of the diverse metrics of quality that could be measured by using ontologies.

4.3 Reuse

Once a project has been created and the ontology build properly, when the next project arrives to the system engineering process, the requirements related to the project in a specific area could be reused as well as the ontology involved for the small set of requirements to be reused.

5 Conclusions

The requirements in any project are one of the most important assets, if not the most. A bad group of requirements might have terrible implications in a developed system. For instance a requirement detailed in various parts of the requirement list using different measurement units might cause an accident or a failure during operation of any system.

Classical sequential review process of requirements is costly in terms of time consuming. Then support of tools for lexical, syntactic analysis enables to correct bad requirements writing before business of project reviews.

One of the next challenges in the Industry is to reach an ontology-assisted system engineering process to write SMART requirements at a first shot.

The use of ontologies and patterns is a promise of doing better requirements engineering and knowledge reuse in any system engineering project, for instance CRYSTAL [2] project that willl apply ontologies and the patterns to facilitate the human job, avoid mechanical checks and increase quality.

References

1. Fraga, A.: Universal Knowledge Reuse. PhD Thesis. Carlos III of Madrid University (2010)
2. CRitical sYSTem engineering AcceLeration (CRYSTAL EU Project), http://www.crystal-artemis.eu/ (last visited November 13, 2013)
3. Braude, E.: Software Engineering. An Object-Oriented Perspective. John Wiley & Sons (2001)
4. Fanmuy, G., Fraga, A., Lloréns, J.: Requirements Verification in the Industry
5. Genova, G., Fuentes, J., Llorens, J., Hurtado, O., Moreno, V.: A framework to measure and improve the quality of textual requirements. Requirements Engineering, http://dx.doi.org/10.1007/s00766-011-0134-z, doi:10.1007/s00766-011-0134-z
6. Guide for Writing Requirements, INCOSE Product INCOSE-TP-2010-006-01, V. 1 (April 2012)
7. Chale, H., et al.: Reducing the Gap between Formal and Informal Worlds in Automotive Safety-Critical Systems. In: INCOSE Symposium 2011, Denver (2011)
8. Alexander, I.F., Stevens, R.: Writing better requirements. Addison-Wesley (2002)
9. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. John Wiley & Sons (1997)
10. Sommerville, I.: Software Engineering. Pearson-Addison Wesley (2005)
11. INCOSE, Systems Engineering Handbook, http://www.incose.org/ProductsPubs/products/sehandbook.aspx (last visited November 13, 2013)
12. Dick, J., Llorens, J.: Using statement-level templates to improve the quality of requirements. In: ICSSEA 2012 (2012)
13. Sowa, J.: Conceptual structures: Information processing in mind and machine, Addison Wesley (1983)
14. OBSE Fundamentals, The Reuse Company, Juan Llorens (UC3M) - José Fuentes (TRC), http://www.reusecompany.com (last visited November 13, 2013)
15. Protégé. Stanford University Development, http://protege.stanford.edu (last visited November 15, 2013)
16. Report CMU/SEI-2008-SR-034
17. Requirements Engineering, PTC Product and Service Advantage, http://www.mks.com/solutions/discipline/rm/requirements-engineering (last visited November 13, 2013)
18. Requirements Patterns for Authoring, MASI, Panagiotis Mourtis & Susana Beltrán. Internal Report UC3M (2012)
19. Pressman, R.: Software Engineering: a practical guide, 6th edn. McGraw-Hill
20. The Reuse Company (TRC), http://www.reusecompany.com (last visited November 13, 2013)

How Can Usage Monitoring Improve Resilience?

Jean-René Ruault, Frédéric Vanderhaegen, and Christophe Kolski

Abstract. Resilience and systems engineering are key issues for critical systems. The operational usage and states of such systems are quite different from reference ones, generating drift and generate risks. This article suggests functional and physical architectures that fit resilience. Four functions relate to resilience (avoidance, resistance, recovery, adaptation). We develop the avoidance one and define a usage monitoring system that implements it. The case study concerns a railway accident that occurred at Aldershot, Canada. We explain the origin of the gap leading to the accident. The usage monitoring system would allow human operators to understand the situation and avoid the accident.

1 Introduction

Nowadays, resilience is a key issue for complex system, with many books and articles dealing with resilience [3], [10], [12], [13], whatever the domain, in order to cope with unexpected events. Systems engineering is another key issue [8], [6]. Many critical and complex systems show a very long lifecycle. We can't foresee all the operational situations that they will meet. The resilience of a system facing unforeseeable events is a critical challenge. The paper suggests a solution to monitor system real states in order to assess drift and to alert human operator of the proximity of hazard. The first part of this paper summarizes the state of the art, for systems engineering and resilience. The second one details a design pattern fit to

Jean-René Ruault · Frédéric Vanderhaegen · Christophe Kolski
Université de Lille Nord de France, 59000 Lille, France - UVHC, LAMIH,
59313 Valenciennes, France, CNRS, UMR 8201, 59313 Valenciennes, France
e-mail: surname.name@univ-valenciennes.fr

Jean-René Ruault
DGA, 7-9 rue des Mathurins, F-92221 Bagneux, France
e-mail: jean-rene.ruault@intradef.gouv.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_12

resilient systems. It contents functional and physical architecture models. It details impacts on the usage monitoring components and on the user interface. The third part applies these concepts to a case study, in the railway domain.

2 State of the Art

The state of the art details the main concepts upon which is based this paper, that are systems engineering, systems architecture, systems modeling language, as well as resilience.

2.1 Systems Engineering, Architecture, SysML

The ISO 15288 standard defines a system as "a combination of interacting elements organized to achieve one or more stated purposes" [6], while IEEE 1220 defines it as "a set or arrangement of elements [people, products (hardware and software) and processes (facilities, equipment, material, and procedures)] that are related, and whose behavior satisfies operational needs and provides for the life cycle sustainment of the products" [5].

For instance, a railway system allows transporting travelers and fret from point to point. Such a system encompasses: end products or services, for instance, travelers transportation, fret transportation; equipments and devices producing these end products and services that are trains, stations, traffic management systems, sale systems; enabling systems, that are CASE tools or test tools; end users specified processes and activities, such as driving train, managing traffic, vending tickets, conforming regulation rules; end users specified profiles, roles and responsibilities, such as engineers, traffic managers, structure organization, that specifies both end users profiles, roles and responsibilities, as well as processes and activities; resources, such as electricity.

The "system in use" is quite different to the "system as designed". End users behave to reach performance goals, control their activities, adapting them function of contextual contingencies and improvement of performance requirements, and resolve system dysfunctions and failures. Most of the security and safety analysis are based upon foreseen and predictable failures and do not take into account unforeseen and unpredictable events. So, the system is not designed to perform in such a way and the users have to resolve the gap between the unpredictable events and the system functions. The system architecture models describe the organization of the functions and the components of the system. Main architectural points of view are:

- The operational one: why is the system designed and built? What are its missions and goals? What are the operational missions in which it will be used?
- The functional one: what are the services that the system provides to its environment? What is the organization of these services?
- The physical one: how the system's components interact together in order to provide these services?

Nowadays, the system modeling language (SysML) provides a set of diagrams in order to elaborate these models [1]. These diagrams allow modeling structure and behavior of a system. Moreover, including requirements diagram, SysML allows traceability between models and requirements. SysML is a key driver of model based systems engineering. This set of diagrams contents: (1) Structure diagrams set (block definition diagram and internal block diagram); (2) Behavior diagrams set (activity diagram, sequence diagram, state machine diagram and use case diagram); (3) Parametric diagram; (4) Requirements diagram and, (5) Package diagram.

2.2 Resilience Functions

Resilience is an extrinsic relational property of a system. It characterizes its property "to cope" with adversity where the disturbance is unforeseeable. Authors of the "Resilience engineering" book [3] states: "Resilience is capacity of a system or an organization to react and recover after a disturbance, with a minimal effect on dynamic stability". Moreover, resilience is complementary and adds value to other system safety method.

Luzeaux [8] characterizes resilience as a "management at the border of the domain of application… The challenges linked to resilience include the management of that which is uncertain or unplanned, accidents, the transition between more or less catastrophic circumstances while avoiding a true catastrophe, and the return to a more normal operational status". Luzeaux [8] differentiates four main resilience functions which complements each other: 1) avoidance (capacity for anticipation); 2) resistance (capacity for absorption); 3) adaptation (capacity for reconfiguration), and 4) recovery (capacity for restoration). We focus now on the first key function for resilience: avoidance [8]. The avoidance function consists of acquiring information at the operators' level in order to anticipate and to avoid the accident, that is: (1) To obtain a representation of the environment; (2) To obtain a representation of the system dynamics: (3) To identify the environment states that were not envisioned; (4) To evaluate the instantaneous or trend drifts; (5) To evaluate the proximity of the state of the system compared to the hazard.

These functional elements impact the architecture of the system of interest, enabling systems architecture, since we give to operators an appropriate situation awareness representation. Resilience is the dynamic process that allows the crew to understand the current situation, to learn and develop adequate behaviors to take into account environment adversities and to adapt as well as possible. It is the capacity of a sociotechnical system to continue to evolve and fulfill its operational mission in spite of the difficult conditions, serious constraints or events, sometimes severe damages or losses. This adjustment capability is based upon the dynamic process of "visual piloting". The system must have a great capacity to estimate its position with regard to the danger zone [8]. The system must be designed to cope with uncertainty. It is necessary to specify the envelope of required, desirable, even acceptable, execution and to require that the system recognizes the

situations where it is likely to leave this envelope. “Resilience is obtained via the system capability to monitor conditions at the edges of the performance envelope, determining their value and the usual distance from the edge and the ability to adapt the operational behavior of the system to potential developments in the envelope…” [8]. The objective is to qualify and quantify the drift of the system towards the state of failure before a major breakdown occurs.

In many cases, the system has been designed to be safe under specified conditions, but there are no means to monitor the system when it operates under unspecified conditions, and to reassess actual risk. Safety under this situation is neither monitored nor controlled. The resilient management consists of clear, relevant and shared situation awareness, among all the communities, which implies to assess the gap between the specified path and the actual one as usual fluctuations or, on the opposite, the trend of a forecast latent deviation.

Hardy [2] expresses that “plans that do not reflect reality may create the impression that effective organization is in place and that risks have been reduced, when in fact large risks could exist”. This difference may grow from the beginning of the operation of the system, from step to step, generating a gap between the plans and the reality. The figure 1 expresses this difference and the gap. The specified path deals with the specified task [7], or the work-as-imagined, taking place along the time. It contains specified local variability included within tolerance margins that is everyday or ‘normal’ variability [4] as defined *a priori*.

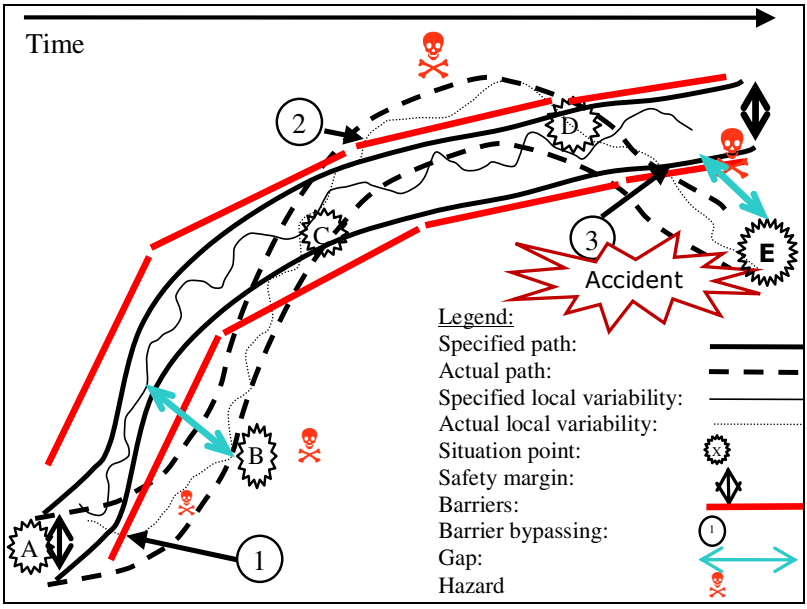


Fig. 1 Specified and actual paths of a sociotechnical system [10]

The actual path, among other possible ones, denotes the actual activity [7], or the work-as-done [4], of the sociotechnical systems, function of met contingencies. This actual path contains actual local variability, since these contingencies are not stable and linear. The gap between these two paths is due to unusual conditions, 'out-of-range' variability [4], that is not an isolated case, but a huge trend. These unusual conditions may be new and unforeseeable working environments conditions. By coming of A, the real dynamics by-passes the barrier in 1 (cf. figure 1), moves towards B, then C, to join D and E by by-passing of new the barrier in 2 and 3. This real dynamics of A in E expresses a gap which can be far from the prescribed dynamics, as it is the case B. Nobody can estimate this gap. Nobody is conscious of situation and can estimate the risk infers by the drift. Stage after stage, the dangerous actions increase the risks (☠), until the accident (E). The critical issue is the capacity of the user interface to give to the operators a shared situation awareness and to allow a navigation at sight. It is necessary to compare the real states and usage of the system and the reference ones.

3 Design Pattern Fit to Resilient Systems

We elaborate a design pattern that fits resilient systems. This design pattern declines the avoidance function in a functional architecture, then in a physical one. We will decline the other functions in further articles. The main functions of the resilience are avoidance, resistance, recovery and adaption. In this article, we detail the avoidance function.

3.1 Functional Architecture: Monitor System's Usage and Current State

The goal of this function is to be aware of the current situation compared with the specified one that is the drift compared with the nominal path, the proximity of hazard, and the safety margins.

It consists of gathering information about the system, its dynamics, its environment, and alerting operator when the system deviates from its nominal path. This function (figure 2) is decomposed into these following four functions on which we focus: (1) To obtain a representation of the system dynamics; (2) To evaluate drifts; (3) To evaluate proximity of hazard and (4) To alert operators. These functions are allocated to components in the physical architecture of the usage monitoring system.

3.2 Physical Architecture: Usage Monitoring Components

These four functions are allocated to the usage monitoring system. It provides two sets of services respectively for these two functions: (1) To obtain a representation of the system dynamics; to gather usage from operating parts of a system; (2) To alert operators; to express warning.

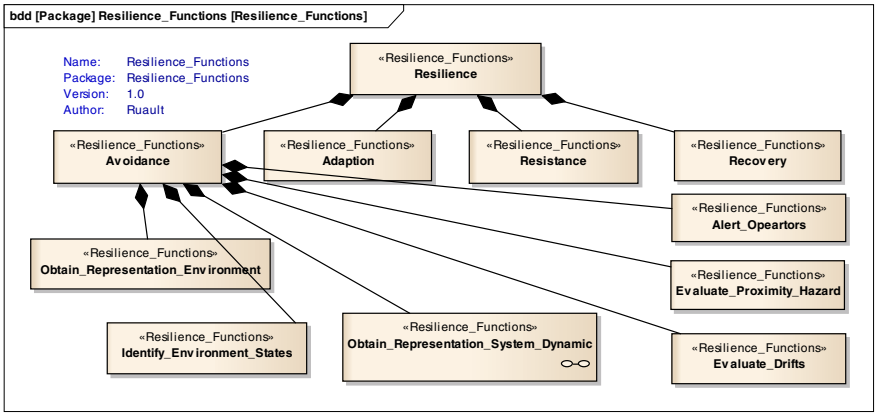


Fig. 2 Functional decomposition of the avoidance function (block definition diagram)

The two other functions are implemented inside the usage monitoring system that encompasses a set of components that are:

- Usage sensor proxies are closely nested to the security devices or other components of the systems, gather their states and usages and send them to the respective usage sensors. Each component and security device that contributes to resilience has a usage sensor proxy that fits to it.
- Usage sensors get the states and usages of devices and components and translate them in order to be analyzed. Each component and security device that contributes to resilience has a usage sensor that fits to it.
- Current state repository stores the data coming from the usage sensors whatever they are, in order to assess trend drifts. It deals with reality [2].
- Reference state repository contains models that specify security, including specified variability, barriers characteristics, as well as more specific data. It deals with plan [2].
- States comparison engine compares the current states and the reference ones, in order to assess drifts and evaluate the proximity of hazard. It sends warning levels, safety margins and drifts to the user interface proxy.
- User interface proxy is closely nested to the other user interface of the system and expresses warning in order to alert human operators.

The table 1 shows the allocation of resilience functions on the usage monitoring system components.

The block definition diagram (figure 3) shows the physical architecture of the usage monitoring system, and each component with its operations and attributes, that have to be tailored in order to fit domain specificities and safety stakes.

Table 1 Allocation of resilience functions on the usage monitoring system components

Functions	Obtain a representation of the system dynamics	Evaluate the drifts	Evaluate the proximity of the system compared with the hazard	Alert operators
Components	Public service	Internal function	Internal function	Public service
Usage sensor proxy	X			
Usage sensor	X			
Current state repository		X	X	
Reference state repository		X	X	
States comparison engine		X	X	
User interface proxy				X

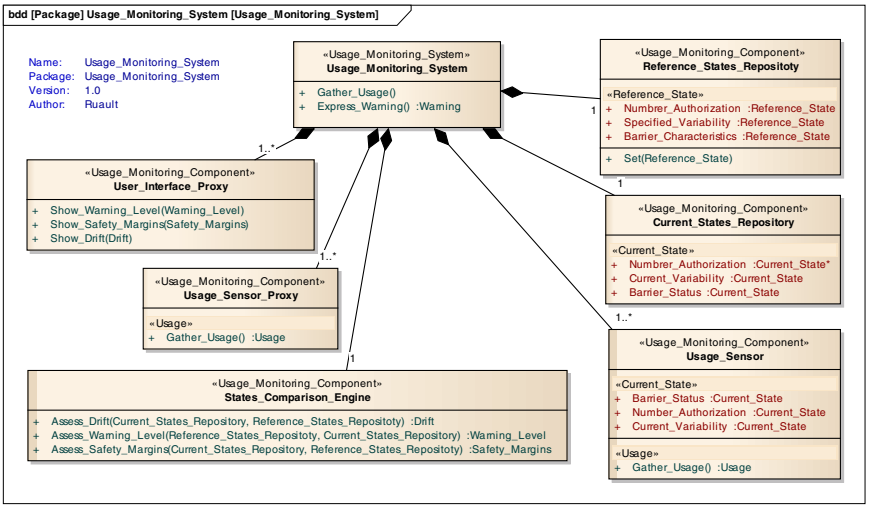


Fig. 3 Physical architecture of the usage monitoring system

3.3 Impacts of the Usage Monitoring Components Upon Functional and Physical Architectures of Whole System

The system architecture must evolve in order to link together the operating system and the usage monitoring system. We differentiate two parts. On the one hand, the operating system implements the core functionalities, that is the part that reach the goals and realizes the operational missions. On the other hand, the usage monitoring system implements the avoidance functions of the resilience. Any kind of systems can implement this pattern, rail traffic management systems as well as trains, tracks or stations (figure 4).

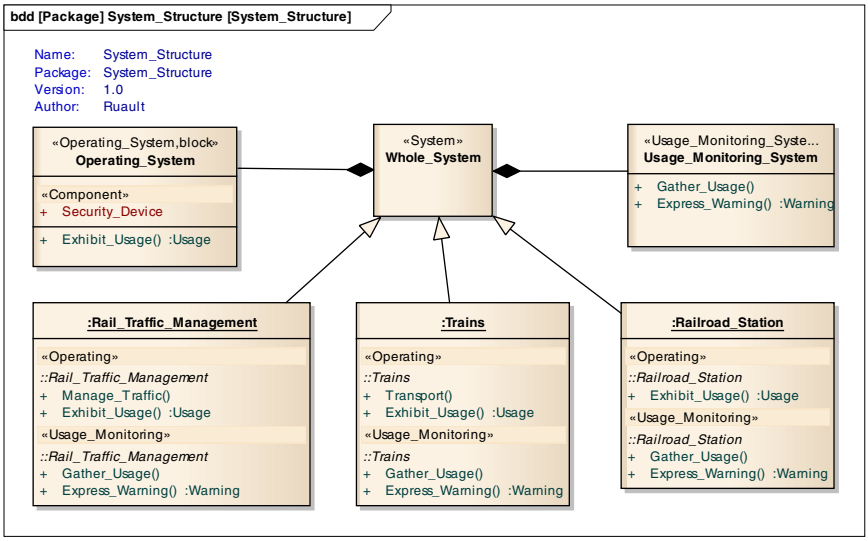


Fig. 4 A whole system containing an operating system part and an usage monitoring system one

For each of these different systems, their operating parts exhibit usage and state, as well as their specific functionalities. On the other side, their usage monitoring parts gather usage and express warning. That implies interfaces and flows between these two types of parts (figure 5). The operating system exhibits usage, for instance current variability, barrier state, among other safety information. The usage monitoring system gathers these information and, function of the real state of the operating system, expresses warning in order to alert the operators, such as safety margins, drift or proximity of hazard.

3.4 Impacts of the Usage Monitoring Components Upon User Interface

The system architecture must evolve in order to express the warning to the operators, via the relevant user interfaces. This relies on the capacity to measure its current internal states and explain the gap between them and the reference ones. These interfaces must be designed in order to express safety margins, hazard proximity and increase of risk level.

So, the operators can regulate their activities, evaluate the differences between the current system situation and the system field of definition, and detect, as soon as possible, the migration or compensation mechanisms. We suggest a solution that needs to be assessed with operators. It expresses the progressive drift from secure situation to risky one [10]. The operators must be able to see that the system is in a high risk zone with catastrophic consequences. Since that the operators are awarded of the proximity of hazard, they can take care, improve procedures and monitor the real state of the system.

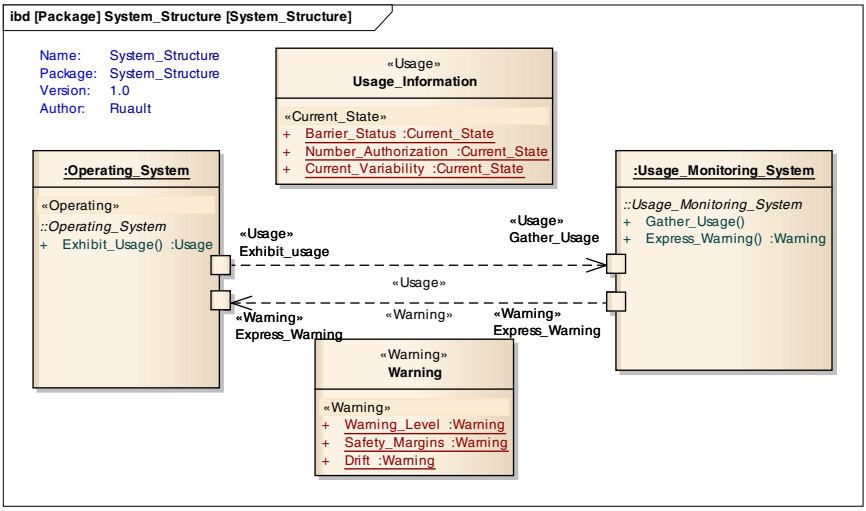


Fig. 5 Interfaces and flows between operating system and usage monitoring system (internal block diagram)

4 Case Study: Railway Accident

The application is the railway accident that occurred at Aldershot station [11].

According to the accident report [11], "On 26 February 2012, VIA Rail Canada Inc. passenger train No. 92 (VIA 92) was preceding eastward from Niagara Falls to Toronto, Ontario, on track 2 of the Canadian National Oakville Subdivision near Burlington, Ontario. VIA 92". The investigations show that while approaching the Aldershot station, the crew encountered a first signal (Clear to Limited) and then another one (Clear to Slow). The signals were specifying to proceed and approach signal 334T2 located east of the Aldershot station at 24 km/h (15 mph). This consecutive information was part of the signal indications governing VIA's 92 movement in order to pass from the track 2 to the track 3. However, the stop at the Aldershot station was an event that interrupted the signal progression. Hence the crew was more preoccupied by stopping the train at the station than proceeding to the signal 334T2 with appropriate speed. During the stop, there was no further indication to remind the crew of the previous signal. This event laid to the interruption of the signal indications, promoting oblivion of the past information. Moreover, in 99 % of the cases, the train of the company circulated on the track 2. This day, works were realized on the track 1 and the track 2. An authorization to occupy the track was granted to the team in charge of these works by the controller of the rail traffic. The team of exploitation of the train was not informed about these works. The train had to pass of the track 2 in the track 3, via a crossover between track 2 and 3. This instruction was communicated by the railway signals which the team of exploitation is supposed to apply scrupulously. The speed on

this connection was limited to 24 km/h (15 mph). The team of exploitation understood too late the situation. The train entered on the connection a 108 km/h speed (67 mph) and went leaned. When a passage of the track 2 to the track 3 is necessary, this passage is realized on a crossover for which the authorized maximal speed is 72.42 km/h (45 mph). The day of the accident, the situation was quite different. The drivers VIA Rail had to pass of the track 2 in the track 3 on a crossover as which the authorized maximal speed was of 24 km/h (15 mph). This situation, the day of the accident, generated an important gap between the specified speed, adapted to pass on the crossover, and the real speed of the train. The instruction of speed was shown on the railway signals, before Aldershot station in which the train stopped. There was no reminder of the specified speed when the train restarted of Aldershot station. The speed of the train was excessive and the capacity to brake insufficient to enter on the points limited in 24 km/h (15 mph). The drivers were not conscious of this gap. When they understood the situation, it was too late. The drivers were not able to avoid the accident.

An usage monitoring system is useful for detecting the major violations of safety. In the case study of the Aldershot station railway accident, a usage monitoring system would have been helpful for the following issues:

- Detecting the speed excess by comparing it to the accepted threshold and presenting the evidence of a speed exceeding to the operators.
- Managing the rail crossing between rail 2 to rail 3 by alerting the operators or presenting a visual device with the maneuver to be accomplished.
- Reporting the railway signals to the operators in real-time in order to keep them informed and secures the situation awareness of the operating crew.

Indeed, from the moment when the VIA 92 entered the crossover No. 5 (1) with excessive speed, consistent with the crew misperception of the railway signals (2), it could be imagined that the usage monitoring system would have reported the anomaly through the usage sensor components. Hence, during the stop at the Aldershot station, interrupting the continuous progression of signals (3) (4) the usage monitoring system might have been helpful to the operators for understanding the situation compared to the reference state (specified path cf. Fig 1) with the state comparison engine component.

Among other possibilities, we suggest an architecture allowing showing to the drivers this gap so that they become aware of the situation and adapt the speed in a appropriate way. The specified speed and speed limited to 24 km/h (15 mph) must be transmitted by the traffic management system in the train, in order to the usage monitoring system can compare these speeds with the real one. The drivers owe informed about the speed limited to 24 km/h (15 mph) of the points which they have to take. That needs to communicate to the drivers of the specified speed, the maximal speed authorized on the points, the gap between the specified speed and the real speed, as well as the necessary distance to brake before to engage on the points.

Both rail traffic management system and train system (figure 6) contents an operating subsystem and a usage monitoring one. The train system gathers specified speed and crossover maximum speed in order to evaluate this information compared to the actual speed and alerts the drivers about the over speed.

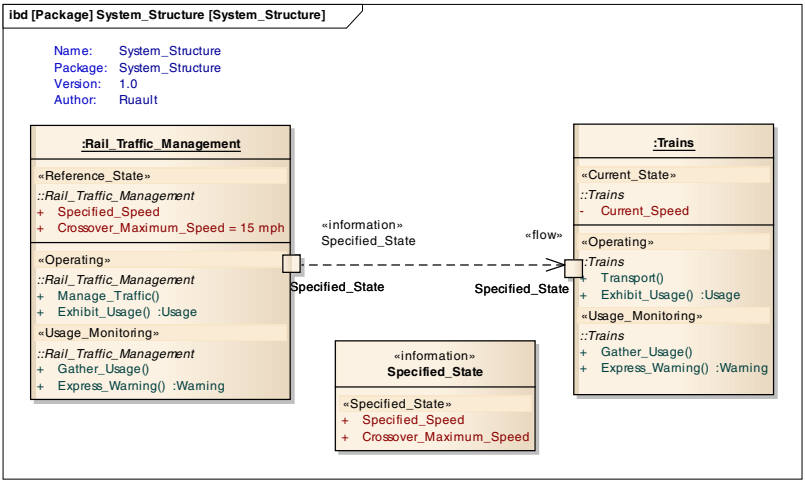


Fig. 6 Communication of specified states from the rail traffic management and the train

Other solutions exist. For instance, a signal after the station informs all drivers of conditions beyond the station, or an adaptive speed limiting based on signals communicates from train to train, in order to prevent collision.

5 Conclusion

Our proposal consists in interconnecting the operating system, which realizes the operational missions, and the usage monitoring system. This interconnection allows monitoring the state, the usage of the system, to estimate the gap between the current state of the system and the safe one, to estimate the proximity of hazard, and to inform the operators. The goal is that the operators share a clear, reliable, relevant and updated representation of the operational context as well as the usage of the system, so that they can take the appropriate measures. The first stage, object of this paper, is to design a system architecture implementing this interconnection to monitor the usage of the system. A second stage will consist in widening this architecture to observe the operational context and express it to the operators. A first difficulty lies in the determination of issues to be observed, in particular the behavior which cannot be *a priori* envisioned. A second difficulty lies in the reliability and the fidelity of the measures. Indeed, the lack of information, or false information, would be error prone and accident prone. Finally, benchmark models allowing estimating this drift have to be available in the system in exploitation, and to correspond to the real configuration of the system.

References

[1] Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML, 2nd edn. Morgan Kaufmann (2011)
[2] Hardy, T.-L.: The system safety skeptic. Author-House, Bloomington (2010)
[3] Hollnagel, E., Woods, D.D., Leveson, N.: Resilience engineering. Concepts and precepts. Ashgate, Hampshire (2006)
[4] Hollnagel, E.: FRAM: The Functional Resonance Analysis Method. Ashgate, Hampshire (2012)
[5] IEEE Std 1220, IEEE Standard for Application and Management of the Systems Engineering Process (2005)
[6] ISO/IEC 15288, Systems engineering — System life cycle processes (2008)
[7] Leplat, J.: Erreur humaine, fiabilité humaine dans le travail, 197 pages. Armand Colin, Paris (1985)
[8] Luzeaux, D.: Engineering Large-scale Complex Systems. In: Luzeaux, D., Ruault, J.-R., Wippler, J.-L. (eds.) Complex Systems and Systems of Systems Engineering. ISTE Ltd. and John Wiley & Sons Inc. (2011)
[9] Ruault, J.-R., Vanderhaegen, F., Luzeaux, D.: Sociotechnical systems resilience. In: 22nd Annual INCOSE International Symposium, Rome, July 9-12 (2012)
[10] Ruault, J.-R., Vanderhaegen, F., Kolski, C.: Sociotechnical systems resil-ience: a dissonance engineering point of view. In: 12th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, August 11-15. IFAC, Las Vegas (2013)
[11] Transportation Safety Board of Canada, Railway Investigation Report R12T0038, Main-track Derailment VIA Rail Canada Inc. Passenger Train No. 92 Mile 33.23, Canadian National Oakville Subdivision Aldershot, Ontario 26 February 2012 (2013)
[12] Zieba, S., Polet, P., Vanderhaegen, F., Debernard, S.: Principles of adjustable autonomy: a framework for resilient human machine cooperation. Cognition, Technology and Work 12(3), 193–203 (2010)
[13] Zieba, S., Polet, P., Vanderhaegen, F.: Using adjustable autonomy and human-machine cooperation for the resilience of a human-machine system, Application to a ground robotic system. Information Sciences 181, 379–397 (2011)

Executable Architectures Using Cuckoo Search Optimization Coupled with OPM and CPN-A Module: A New Meta-Architecture Model for FILA SoS

Siddhartha Agarwal, Renzhong Wang, and Cihan H. Dagli

Abstract. Understanding System of Systems (SoS) requires novel ways to apply systems engineering processes. Acknowledged SoS have recognized objectives, a designated manager and resources for the SoS. The goal of this research is to develop a proof of concept tool suite for Acknowledged SoS systems simulation. This suite is named flexible, intelligent and learning architectures for System of Systems (FILA-SoS). FILA-SoS assists the SoS manager in architecture generation, selection, and implementation working as an aid for decision making. Binary cuckoo search constrained optimization is used to generate meta-architectures which are evaluated by a fuzzy assessor for quality assurance. The architecture is then converted into an executable structure using Object Process Methodology (OPM) and Colored Petri Nets (CPN). A hybrid methodology comprising of OPM and CPN approach is implemented for simulating the acquisition environment. Initial application for a Search and Rescue (SAR) SoS, consisting of 25 individual systems with ten capabilities gave promising results.

Keywords: Acknowledged, Cuckoo Search, CPN, fuzzy, SAR, meta-architectures, negotiation, OPM, SOS, binary, optimization, simulation.

1 Introduction

Flexible models and scalable search approaches are needed to manage the design space of systems of systems (SoS) architecture, capture its emergent behavior, and assist in dealing with the rapidly changing machinery. Four types of SoS that appear in literature are directed, collaborative, virtual and acknowledged. Acknowledged SoS share properties with both collaborative and directed SoS [1].

Siddhartha Agarwal · Renzhong Wang · Cihan H. Dagli
Engineering Management and Systems Engineering Department,
Missouri University of Science and Technology, Rolla, Missouri, USA
e-mail: {sa265,rwkb4,dagli}@mst.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_13

Acknowledged SoS have documented objectives, a nominated manager, and resources for the SoS. The constituent systems preserve their autonomous ownership, objectives, funding, and development and sustainment approaches. All systems have to work in collaboration to achieve larger goal which they are incapable of achieving just by themselves. Simulation and modeling techniques for Acknowledged SoS are still in their infancy. Our objective in this paper has been to present a module within the larger model called as FILA-SoS that stands for flexible, intelligent and learning architectures for system of systems. FILA is an integrated tool suite in ANYLOGIC* and involves meta-architecture generation for acknowledged SoS using computational intelligence based on key performance attributes specified by the stakeholders [2]. The meta-architectures are a crude form of the final architectures. To realize them in actuality SoS coordinator has to negotiate his demands and resources individually with participating systems [3]. This major issue is resolved by introducing negotiation modules between individual systems and SoS manager based on domain specific information. For further details please refer to the system engineering research center report [4].The set of issues being negotiated over are defined by the domain of the problem. This paper proposes simulation for ISR (intelligence, surveillance and reconnaissance) scenario. The scenario depicts an area pacification/low level guerilla war situation. The garrison has the systems detailed in section 4 for performing ISR and fire support to patrols and convoys and the area being controlled is about 100 nmi in radius. This SoS includes negotiation attributes as price (value for capability being acquired), performance (task execution capacity) and deadline (delivery date). Executable architectures are generated using a hybrid of Object Process Methodology (OPM) and Colored Petri Nets (CPN) [5]. These executable architectures are useful in providing the much-needed information to the SoS coordinator for assessing the architecture quality and help him in negotiating better. The intended contribution of this paper is as follows. First, a different method of meta-architecture generation based on cuckoo search algorithm combined with fuzzy inference engine is proposed for system architecting. Secondly the non-dominated solutions generated, all of which represent a different architecture, are then made executable through the use of OPM and CPN. Finally, our attempt is to present an integrated acknowledged SoS architecting model called FILA-SoS whose capabilities include extensive multi-level SoS meta architecture generation covering the entire design space, flexible and robust architecture assessment, and final architecture securement through simulated negotiations. This paper is motivated by current lack of understanding of system participation choice on the overall SoS capability. In addition, there is an imminent need for domain independent SoS architecture generation and assessment methodology.

The rest of the paper is organized as follows. Section 2 details related work in other SoS projects and FILA-SoS. Section 3 describes the FILA-SoS model used in this study by detailing different modules and some basic concepts. Section 4 describes our methodology for architecture generation and converting them to executable architectures. In section 5, we analyze the results of the proposed procedures. Finally, Section 6 outlines the conclusions and our plans for future study.

2 Literature Review on Current and Past SoS Projects

In this section a brief description of major SoS projects currently being pursued in a variety of domains are discussed. This section will help the reader get an overview of the scope of research being conducted. The descriptions do not necessarily follow any order in which the projected came into inception. DANSE SoS stands for Designing for Adaptation and Evolution in System of Systems [6]. DANSE project addresses the challenging technical, management, and political problems within organizations. The main features include combining the strengths of several infrastructures and objects present because of advances in communications, sensors and actuating competencies. DANSE is among several projects in SoS funded by the European Commission as part of the Seventh Framework Program. The purpose of the DYMASOS (Dynamic Management of Physically Coupled Systems of Systems) project is to explore methods for the distributed management of large physically connected systems along with distributed autonomous management and global coordination [7]. COMPASS stands for Comprehensive Modelling for Advanced Systems of Systems and aims to develop collaborative research on model-based techniques for developing and maintaining SoS [8]. For example, a flexible and responsive SoS can be developed for emergency management, given the fact that individual systems were not intended for collaboration. T-AREA-SoS (Trans-Atlantic Research and Education Agenda on Systems of Systems) was developed through cooperation between EU-US Systems of Systems (SoS) research [9]. T-AREA-SoS aims to achieve European competitiveness and improve the societal impact through development and management of large complex systems. The CYPHERS project aims at developing an integrated cyber-physical roadmap and strategy for Europe [10]. Its ultimate goal is to combine and expand Europe's capability in embedded and mobile computing as well as in control of networked embedded systems. Some projects that are closely related to CYPHERS are Hycon2: highly-complex and networked control systems, EMSIG: embedded systems special interest group, artist design: European network of excellence on embedded systems design and CPSoS: cyber-physical systems of systems. AMADEOS aims critical systems certification for SoS [11]. Its abbreviation stands for Architecture for Multi-criticality Agile Dependable Evolutionary Open System of Systems. The AMADEOS project emphasizes on evolution, emergence, dependability and security, taking into consideration-embedded devices and the cloud as the projects execution platform. It has three significant objectives namely: to introduce a concept of global time that can be accessed and recognized by all elements of the SoS, ability to explain and formalize SoS evolvability and dynamicity, and handling emerging properties in SoS. The CPSOS is a support action, to be completed in 30 months, that aims at developing a roadmap on research and innovation in engineering and management of cyber-physical systems of systems [12]. CPSOS are cyber-physical systems which exhibit the features of systems of systems. The aim of CPSOS is to study and analyze computing and communication systems that interact with large complex physical systems. Local4Global- project stands for Systems of Systems that act locally for optimizing globally [13]. Its desired goal is to develop, comprehensively test and evaluate in real-life Traffic Systems of Systems (TSoS). In

addition, the project needs to generate a generic, integrated and fully functional methodology for TSoS. The optimization method developed so far is demonstrated in two real scenarios: the climate control of a building and optimizing the traffic on a test site in the North of Munich. Another traffic prediction project involving SoS techniques is smarter traffic predictions in collaboration with IBM for the city of Cologne, Germany. COBWEB - Citizens OBservatory WEB – is another project that is funded under the European Union's Seventh Framework Programme (FP 7) for developing community-based environmental systems using innovative and novel earth observations applications [14]. It is a large collaboration of experts from 13 partners and 5 countries. The projects major aim is to create a platform environment enabling citizens living under the biosphere reserves designated by UNESCO (United Nations Educational, Scientific and Cultural Organization) to collect environmental data using their mobile devices. GEOSS stands for global earth observation system of systems, aims to provide solutions for a number of problems around the world [15]. So far, it has been used in forecasting meningitis outbreaks, guarding biodiversity, and helping in improving climate observations in Africa and Central and South America. The environmental protection agency (EPA) in USA along with Group on Earth Observations (GEO) helps in contribution to the development of GEOSS. The ultimate goal of GEOSS is to provide decision makers with correct and prompt scientific information for advancement of social benefits. Integrated Mobile Security Kit (IMSK) aims at detecting critical situations [16]. It helps to provide quickly an effective deployment of information fused with intelligence on mobile platforms for enhanced security. Some examples of its application are mass events such as football games and terrorism attacks. Lastly, the ministry of economics and technology in Germany sponsors Shared e-Fleet project [17]. It aims at higher utilization of systems electric vehicles so that they can be used commonly and very efficiently.

2.1 Recent Work on FILA-SoS

Some of the recent work that has been done in FILA-SoS includes creating meta-architecture consisting of a set of systems and their interconnections using binary particle swarm, genetic algorithms and modular type II fuzzy nets as fitness evaluator [18]. Executable systems architectures are created using CPN models and their simulations contain detailed quantitative information about the performance of a system, such as throughput and processing time. These results support the exploration and detection of structural and dynamic system properties [19]. A bilateral counter offer based negotiation mechanism between SoS manager and individual system is employed involving three attributes namely performance demands from each system, deadline to prepare for participation in SoS, and amount of funds to be received as remuneration from the manager. The system behaviors are anticipated to be basically of three types selfish, cooperative and opportunistic. ANYLOGIC [20] based integrated model is designed to incorporate negotiation strategies and generate multiple responses of the counter offer game between the two parties. A number of negotiation rounds with different system types and SoS coordinator are conducted [21]. This negotiation data is then

analyzed through various unsupervised clustering approaches to delineate behaviors. An unsupervised clustering of the difference between offer and counteroffer of attributes by hierarchical and k-means techniques reveal four optimal numbers of clusters present in the data [22].

3 Flexible Intelligent & Learning Architecture for System of Systems-FILA-SoS

The authors propose an integrated model that provides a useful multidisciplinary tool for modeling and simulation of systems of systems in an acquisition environment. FILA-SoS examines, the different mathematical models working in cohesion, to determine SoS characteristics that can help increase the chance of success of complex SoS development efforts. In this disposition, this section presents the following concepts and related mathematical models. The initial inputs to the model include selecting a context and environment for the model. The meta-architecture generation ingredients also include a set consisting of participating systems and their interconnections. Figure 1 shows a process flow diagram of the modeling and simulation of SoS. Multiple negotiation rounds and assessment for the architecture after each negotiation round is termed as an epoch in the cycle. Whereas once the architecture selected after multiple epochs is called the final architecture ready for implementation. This culminates a wave of architecting methodology. The next wave would involve different set of stakeholders, new systems and new capabilities being added to the inputs for meta-architecture generating process. Some systems will be excluded and thereby reducing some capabilities of the SoS.

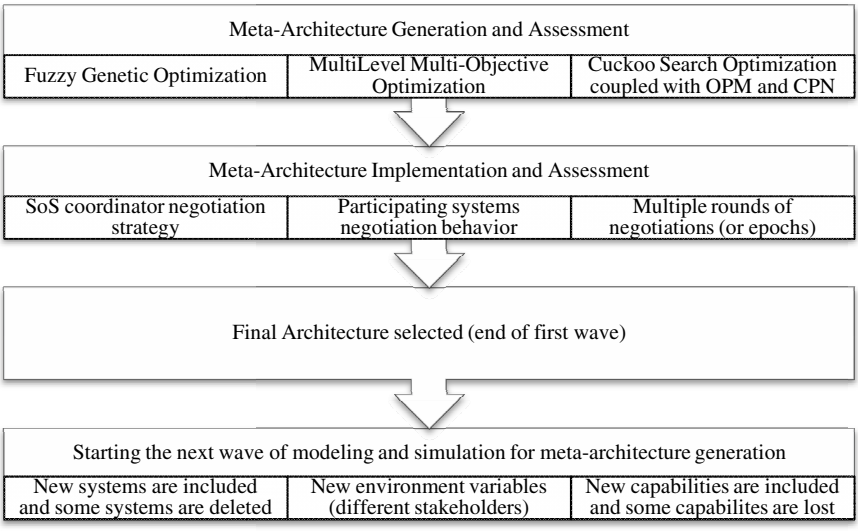


Fig. 1 Process flow of modeling and simulation technique adopted for FILA-SoS

3.1 FILA-SoS Model Overview

The SoS achieves the required capability goal by combining existing system capabilities and adding new capabilities. The integrated system model is comprised of three main elements: SoS acquisition environment, SoS behavior, and Individual system behavior. The following sections present a brief overview of different modules contained within the FILA-SoS.

Meta-Architecture Generation. The SoS meta-architecture includes:

- The systems selected.
- The interfaces selected amongst the systems.
- The capabilities possessed by the systems

The three meta-architecture generation models are: multi-level optimization model, fuzzy-genetic optimization model and hybrid particle swarm-fuzzy modular nets model. This paper proposes a new meta-architecture generation methodology based on hybrid binary cuckoo search [23, 24] constrained optimization algorithm and fuzzy logic for architecture generation.

Architecture Assessment Models. These models are based on multiple stakeholders' point of view and are domain specific. The fuzzy assessor is used to evaluate the fitness of architecture. Fitness function is an amalgamation of multiple key performance attributes (KPAs) of architecture, such as–affordability, performance, robustness, modularity, net-centricity. These KPAs are developed through guided discussions with stakeholders and subject matter experts (SMEs). The attributes will be domain adjusted and selectable, using guidance from SMEs. Fuzzy membership functions (derived from stakeholder views) describe the degrees of goodness in each attribute area. Fuzzy rules (also derived from stakeholder views) combine attributes into an overall fitness grade. The two types of assessors in FILA-SoS are:

- Fuzzy Type I assessor
- Fuzzy Type II modular nets

Acknowledged System of Systems Negotiation Model. After generating meta-architectures, for their actual implementation there is a need to model negotiations between individual systems and the Acknowledged systems coordinator. The model assumes that there will be bilateral negotiation between individual systems and the SoS coordinator. The idea is to assist Acknowledged SoS coordinator in forming a joint capability. This module provides a contract and convinces the individual systems to join the SoS architecture. The SoS coordinator motivates individual systems to do their tasks well. The SoS negotiation model is based on incentive contracting mechanism inspired by game theory.

Negotiation Protocol. The steps in order of occurrence of the protocol are listed below:

1. Send connectivity request to individual systems
2. Individual systems generate counter-offer values based on its behavioral characteristic
   - Selfish
   - Opportunistic
   - Cooperative
3. SoS manager utilizes the incentive based negotiation model to find the optimal strategy for the SoS manager
   - Accept
   - Counter-offer

Individual System Behavior Models. In an interoperable world, reasoning about counterpart's behavior improves the predictive power of the opponent that is negotiating. The individual systems negotiation strategy is described by three behavioral traits namely selfish, cooperative and opportunistic. Three behaviors that are exhibited by the individual systems while they negotiate are selfish, opportunistic, and semi-cooperative models. A brief outline for each of the models is given below:

–Semi-cooperative fuzzy negotiation: This model of semi-cooperative behavior is based on an agent's preferences and the negotiation length. Each systems agent has two inherent behaviors of cooperativeness which in is referred to as Purposive behavior and behavior driven by unforeseen circumstances which is referred to as Contingent.
–Selfish Linear Optimization: The necessary condition for an individual system to collaborate with the SoS is to obtain nonnegative incremental profit (hence, termed selfish).
–Opportunistic Markov chain: Opportunistic model allows the system to behave selfishly as well as unselfishly (or selflessly) by tweaking certain tunable parameters.

SoS Coordinator Adaptive Negotiation Strategy. The authors propose the use of deep belief networks as an unsupervised clustering technique for grouping the negotiation responses from the systems.

Executable System Architecture Modeled by OPM and CPN. A hybrid methodology comprising of object process methodology (OPM) and colored petri nets (CPN) approach is applied for simulating the acquisition environment. Aiding the SoS manager in future decision making through adaptive learning of systems behavior.

4 Methodology of Generating Executable Architectures

The domain specific information required for the architect to proceed is mentioned as a list:

- Constituent system capabilities required are numbered as c_i: i= {1, 2,…, M} .
- Total number of systems that are present is represented by S_s: s = {1, 2,… N}.
- A systems capability i is represented as $c_i^{S_s}$.
- A systems amount of area coverage is represented by A^{S_s}.
- The systems operating time is Ot^{S_s}.
- The systems frequency of information collection is I^{S_s}.
- The systems communication capacity is CC^{S_s}.
- The systems time taken to recover between operations is T^{S_s}
- The system type is denoted by an alphabet: {A, B, C, D, E, F, G ,H ,I}

The following equations describe the formulation of multi-objective problem with variables and constraints.

$$\mathrm{f(x)^T} = \{\ \mathrm{f_1(x)}\ \ \mathrm{f_2(x)}\ \ldots\ \ \mathrm{f_p(x)}\ \} \tag{1}$$

$$\mathbf{A(x)^T} = \{\ \mathrm{a_1(x)}\ \mathrm{a_2(x)}\ \ldots\ \ \mathrm{a_q(x)}\ \} \tag{2}$$

$$\mathbf{x^T} = \{\ \mathrm{x_1}\ \ \mathrm{x_2}\ \ \ldots\ \ \mathrm{x_n}\ \} \in \mathbf{X} \tag{3}$$

***x**: vector of the variables; **f**: objective function(s); **a**: inequality constraints;*

The four objectives in this multi-objective problem are amount of area coverage, operating time, communication capacity, and time taken to recover between operations. These are also the key performance attributes of the SoS and are measured as follows: $A^{\mathrm{SoS}} = \sum_{s=1}^{N} A^{S_s}$, $O^{\mathrm{SoS}} \sum_{s=1}^{N} A^{S_s}$, $CC^{\mathrm{SoS}} = \sum_{s=1}^{N} CC^{S_s}$, $T^{\mathrm{SoS}} = \sum_{s=1}^{N} T^{S_s}$. There are a total of eight linear constraints which make sure that at least one systems of each type is selected for forming the overall architecture. There are five systems of type A called shadow which are tactical unmanned aircraft system (UAS), two systems of type B called gray eagle which are UAS as well, type C define apache helicopters and there are two of them. Similarly, there are two system of type D representing command and control surveillance units, type E has two exploitation control systems, type F has one artillery based system, type G, H, I include four, four, and three systems each and they are comprised of voice/chat systems, line of sight capability and beyond line of sight capability systems.

The first part of the methodology screens the initial list of all systems to a smaller subset. This subset will be combined with OPM and CPNs to produce executable architectures.

A binary evolutionary algorithm called cuckoo search (CS) with multiple objectives is used to solve for obtaining the right set of systems. The objectives here are to maximize area of coverage, maximizing operating time, maximizing the communication capacity and finally minimizing the time between operations. The numbers of parameters necessary to be regulated in CS are much less than genetic algorithms (GA) and particle swarm optimization (PSO) making it more appropriate for an extensive class of the optimization processes. For multi-objective CS constraint optimization problems with K different objectives, the rubrics are stated below:

- Each cuckoo lays an egg at a time, and dumps them in a randomly chosen nest. Nth egg corresponds to the solution in the Nth nest.
- The best nests with high quality of eggs (solutions) will carry over to the next generations.
- The quality of the egg is judged by a fuzzy assessor which incorporates rules for all objectives relating to the overall solution quality.
- A penalty term is subtracted from the overall fitness value of the solution. This penalty term is composed of the sum of all the linear constraints. The concept is that the fitness function decreases as degree of the constraint violation increases.
- Each nest will be abandoned with a probability p and a new nest with K eggs will be built, according to the similarities/differences of the eggs.
- Dimension of each solution is equal to the number of variables in the problem
- A rule of thumb recommends to have an initial population possibly more than 2*number of variables*number of objectives.

4.1 Multi-objective Optimization

The solutions are initially represented as a vector of random numbers and using a sigmoid function is converted to binary value. Each solution is assessed by a fuzzy assessor [20] which helps in reducing the complexity and computational time. Out of 16, some rules created to define the trade-offs between the many objectives are stated:

- If operating time is *low* and coverage is *small* and communication is *short* and time between ops is *excess* the SoS architecture quality is *worst*.
- If operating time is *high* and coverage is *more* and communication is *excellent* and time between ops is *average* the SoS architecture quality is *finest*.
- If operating time is *moderate* and coverage is *medium* and communication is *mediocre* and time between ops is *average* the SoS architecture quality is *ordinary*.
- If operating time is *low* and coverage is *small* and communication is *short* and time between ops is *less* the SoS architecture quality is *finest*.

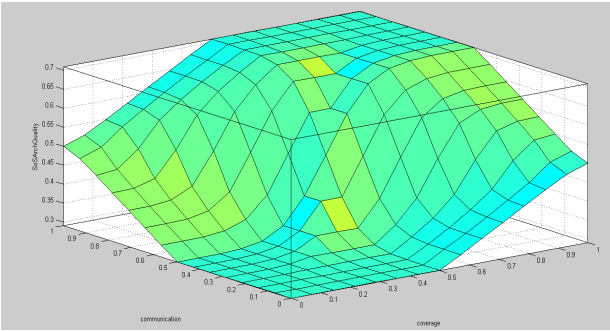


Fig. 2 A plot of the non-linear output surface of a given fuzzy inference system (fis) using the communication and coverage as inputs and the SoS Architecture quality as the output.

Table 1 A list of 6 solutions presented as binary vectors*

A					B		C		D		E		F	G			H						I
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	1	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

* "1" system present and "0" means system absent

The colors highlight the type of each system. The architecture quality of the solutions obtained differed from each other only in the second decimal. The highest quality obtained was 8.31.

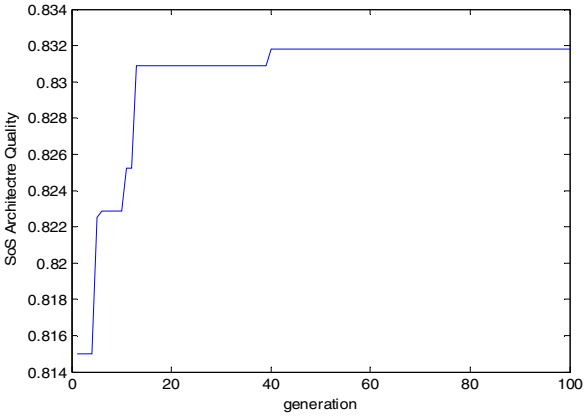


Fig. 3 Graph of architecture quality changing with each generation of population

4.2 Executable Architectures

A more detailed assessment of the performance of the SoS to be built entails the consideration of the interactions between the participating systems in achieving the overall SoS capabilities. This include examining the work flow, the dependency between functions and capabilities, the information or resources flow between related activities, and parameters that define these interactions.

To facilitate such analysis, an executable architecture model is imperative. In this research, a modeling approach that combines the capabilities of OPM and CPN is proposed. Specifically, OPM is used to specify the formal system model as it can capture both the structure and behavior aspects of a system in a single model. CPN supplements OPM by providing simulation and behavior analysis capabilities. Consequently, a mapping between OPM and CPN is needed. Such mapping can follow the one developed in [25, 26]. OPM modeling supports both object-oriented and process-oriented paradigm. CPN supports state-transition-based execution semantics with discrete-event system simulation capability, which can be used to conduct extensive behavior analyses and to derive many performance metrics. The incorporation of CPN also allows the developed system model to be doubled as an analysis model. A large collection of analysis methods and tools developed for CPN can be utilized for strong model analysis, verification, and validation. This modeling approach is explained here with ISR system development as an example.

Problem Definition. Suppose an ISR system to be developed is required to work under either of the following two scenarios in carrying out its designated mission: Scenario 1: Find concentration and movement of irregular troops that might be a warning of impending attack; and Scenario 2: Provide surveillance support to friend troops during an attack. The systems participating the SoS need to collaborate with each other to implement such scenarios, which can be described, in more details, by following work flow:

1) The C&C selects and dispatches required number of systems that carry EO, IR or SAR capabilities to the target terrain.
2) The deployed systems carrying EO, IR or SAR capabilities watch the designated terrain and collect the observations with the carried equipment.
3) The collected data are then transmitted over wide-band data links (provided by communication systems) to the exploitation centers in real time.
4) Upon receiving the data, the exploitation centers analyze the data and provide recommendations.
5) The recommendations are sent to the C&C through the communication systems.
6) C&C makes decisions based on the data received from the exploitation center. Such decisions may include adjusting the deployment of the EO, IR and SAR capabilities by sending command through communication system to those deployed systems.

Table 2 summarizes the related parameters of these systems the capabilities each of the system carries.

Systems	*Shadow*					*Gray*		*apache*		*C&C*		*Exp.*		*Atl*	*Voice/Chat*				*LOS*				*BLOS*		
	A					B		C		D		E		F	G				H				I		
ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Operating time (hour)	*9*	9	9	9	9	25	25	2	2	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
Time between Operations	*5*	5	5	5	5	24	24	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Coverage	*85*	85	85	85	85	150	150	200	200	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Capacity	-	-	-	-	-	-	-	-	-	10	10	10	10	40	20	20	20	20	15	15	15	15	10	10	10
Capability																									
a EO	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b IR	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c Radar	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d Command, Control	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e Data Analysis	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
f Fusion	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
g Strike	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
h LOS	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0
i BLOS	0	0	0	0	0	1	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	0	1	1	1

* "1" means possession of capability and "0" means absent of capability

System Model. based on the problem definition presented in last section 4, an OPM model specifying the ISR system can be the one shown in Fig. 3a and Fig. 3b. Fig. 3a presents the overall system model whereas Fig. 3b presents an unfolding of an object, `ReconnaissanceSys`, to reveal its detailed characters. Similar unfolding for other objects in Fig. 3a is omitted in this paper. This OPM model is a class model, from which instance models can be created. The systems identified in Section 4.2.1 are grouped into five types: Reconnaissance, Communication, Exploitation center, Command & Control Systems, and Attack Systems. They are modeled as OPM objects (represented as rectangular boxes). Reconnaissance systems have two states, `Operation` and `OffDuty`, represented as smoothed rectangles encapsulated in the `ReconnaissanceSys` object. Two process (oval shaped) `GoOffDuty` and `GoOnDuty` can change a reconnaissance system from Operation state to `OffDuty` state or vise verse. An OPM object, `Data`, representing all types of data and information to be processed by the SoS, is created on Fig. 3a with 5 states (`Raw`, `ReceivedByExpl`, `Recommendations`, `RecivedByCC`, and `ActionPlan`), each of which represents a stage of the information being

processed during the operation of the SoS. Four processes (`Transmit1`, `Analyze`, `Transmit2`, and `DecisionMaking`) cause the change of states of a data object. The execution of each of these processes requires the support of one or more objects as indicated by instrument links of OPM (circle end connectors between objects and processes).

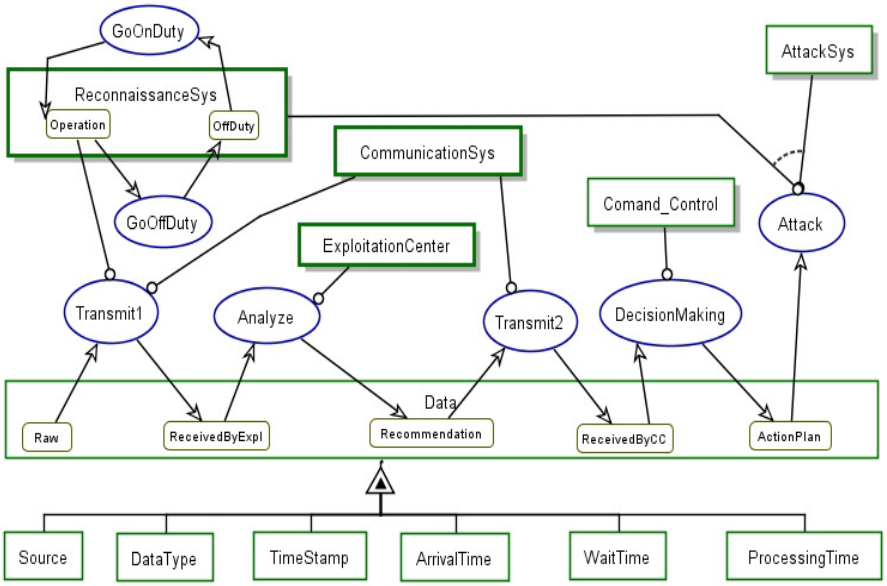


Fig. 4a Generative class model of the SoS represented using OPM

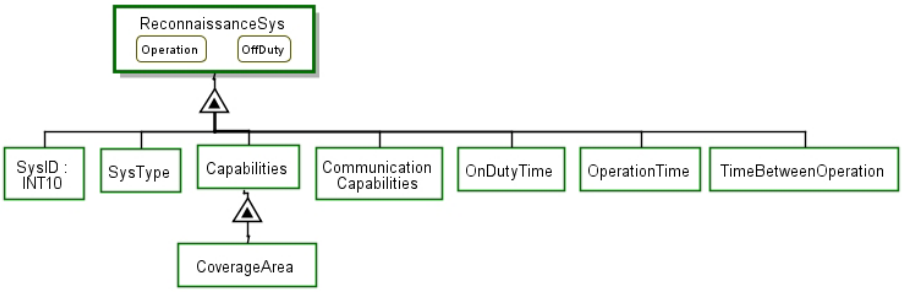


Fig. 4b Unfolding the Reconnaissance System

From such OPM model, A CPN model as the one shown in Fig. 4a and Fig. 4b can be developed by following the mapping rules [25, 26]. The basic idea that maps OPM to CPN is as follows: Map OPM processes to CPN transitions. Map OPM attribute objects (objects connected to their parent object using exhibition-characterization links) to CPN color sets. Such color set thus defines the set of

class attributes for the OPM objects being connected by those attribute objects. Map non-attribute objects that have no states and object states of OPM to CPN places. Map the value(s) of an OPM object to CPN token(s). One or a set of tokens on a CPN place represents either the existence of an object or an object being at the state represented by that place. The former corresponds to the case that the place is mapped from an OPM object with no state and the token(s) on that place represents alternative objects. The latter corresponds to the case that the place is mapped from an OPM state. An object, as in object-oriented modeling, is defined by three parts, states, attributes and services (or methods, functions, or processes). OPM structural links that have no effect on the system dynamics are not mapped to CPN.

In the CPN model shown in Fig. 4a and Fig. 4b, tokens are used to represent the instance of a system in the SoS. Particularly, tokens at CPN place `S_dn` represent the available reconnaissance systems at off duty state, which is also the initial state of all such systems. Such tokens are defined be a list type of value (color set), which encode the attributes that define the corresponding objects. The first element in the list is the system ID (ranging from 1 to 25), the second element is system type (ranging from *A* to *I*), the third element is a list of capabilities carried (ranging from *a* to *i*), the forth element is a list of communication capabilities (with value of *i* and *j*), and the fifth to seventh elements are the time attributes, i.e., on duty time (the simulation time at which a system goes on duty), operation time (mins), and break time (mins), respectively. Tokens at place `Comm` represent the communication systems and they are defined with similar list values. The number of tokens with a particular system ID at that place represents the capacity of that communication system. Similarly tokens and their numbers at place `Exp` represent the exploitation centers and their capacities, respectively. The processing time required by Exploitation center and C&C is simulated by time delays added to the output arc of the `Process` and `Decision` transition, respectively. Such time delays are assumed to follow exponential distribution, the value of which is computed through a function (not shown in Fig. 4a or Fig. 4b). The CPN page `Ltran` models is a substitution transition that represents the data collection activity of the reconnaissance systems. The arrival of data items is assumed to follow an exponential distribution too.

Performance Assessment. The optimization objectives introduced in the beginning of this section can be evaluated through the information collected from running the CPN simulations. The total coverage area $A^{SoS}$of the SoS is calculated from summing up the coverage areas of the individual systems on duty. To get the information regarding what system (s) is on duty at a particular time, a CPN place, `Uact`, is added to the CPN model shown in Fig. 4a, as an output place of transition `Rd`. by collecting the tokens recorded at place `Uact`, it can be known what systems are on duty at a particular time and therefore, the total coverage area of the SoS as a function of time. Accordingly, the average coverage area and minimum coverage area during the entire operation time can also be obtained.

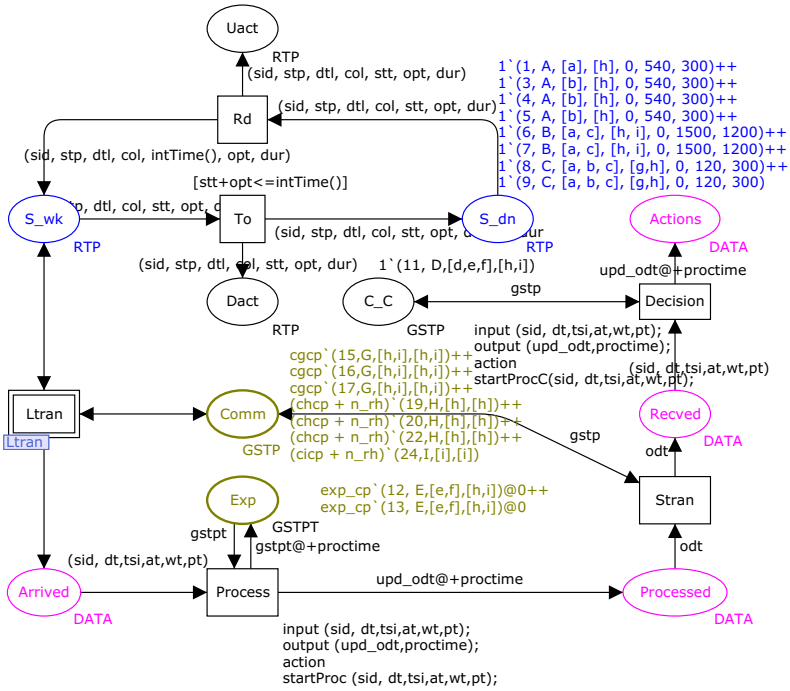


Fig. 5a CPN model converted from OPM

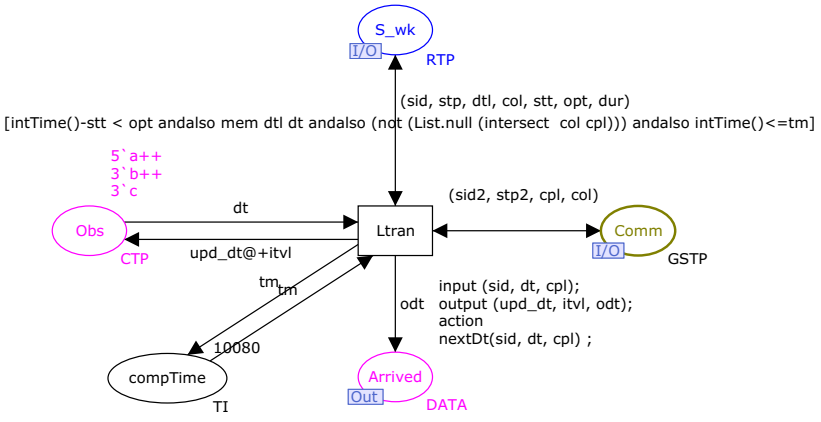


Fig. 5b CPN substitution transition representing data collection activity

The Operating time, time taken to recover between operations can be obtained directly from the on duty/ off duty recorded store at the tokens at place `Uact`. The communication capacity is the sum of individual communication system's communication capacities. In addition, the `Data` token has a list type of value, (`sid`, `dt`, `tsi`, `at`, `wt`, `pt`), where `sid` is the system type identifier, `dt` is the data type identifier (each data type binds to a particular capability, i.e., EO, IR, or SAR), `tsi` is the system simulation time, `at` is the initial arrival time of a data item, i.e., the time that the data is collected by a reconnaissance system, `wt` is the total waiting time, up to the current system time, that a data item spent, and `pt` is the total processing time, up to the current system time, that a data item spent. The difference between the current system simulation time `tsi` and the initial arrival time at is the latency (equals to `wt + pt`). Finally, the total simulation time is controlled by the token value at place `compTime`. It has an initial value of 10080, which represents the period of a week length.

5 Conclusion

The Acknowledged SoS systems can be used to model a broad variety of complex systems in the real world. The ANYLOGIC based integrated model developed in this research is designed to incorporate negotiation strategies and generate multiple responses of the counter offer game between the two parties [27]. The meta-architecture generated is negotiated for possible implementation by the acknowledged SoS. The Binary cuckoo search constrained optimization offers an efficient alternative for generating and selecting optimum meta-architectures. Any one of the architectures can be converted to executable models using the aforementioned techniques. The results thus obtained will help the SoS coordinator in selecting the architecture and negotiate effectively with the systems. The incorporation of OPM and CPN modeling allows the capturing of the interactions between participating systems and the behavior analysis based on such interactions. Accordingly, more detailed performance assessment can be carried out based on such models. Together, these technologies enrich the FILA with comprehensive decision support capabilities.

Acknowledgments. This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

References

1. Dahmann, J., Rebovich, G., Lowry, R., Lane, J., Baldwin, K.: An implementers' view of systems engineering for systems of systems. In: 2011 IEEE International Systems Conference (SysCon), pp. 212–217 (April 2011)
2. Dagli, C.H.: Flexible and Intelligent Learning Architectures for SoS (FILA-SoS) SoS Analysis part 2. In: 5th Annual SERC Sponsor Research Review, Georgetown University, Washington, DC (February 2014)
3. Agarwal, S., Pape, L.E., Kilicay-Ergin, N., Dagli, C.H.: Multi-agent Based Architecture for Acknowledged System of Systems. Procedia Computer Science 28, 1–10 (2014) ISSN 1877-0509.
4. Dagli, C.H.: An Advanced Computational Approach to System of Systems Analysis & Architecting Using Agent-Based Behavioral Model, SERC (2013)
5. Wang, R., Dagli, C.H.: Executable System Architecting Using Systems Modeling Language in Conjunction with Colored Petri Nets in a Model-Driven Systems Development Process. Systems Engineering 14(4), 383–409 (2011)
6. Arnold, A., Boyer, B., Legay, A.: Contracts and Behavioral Patterns for SoS: The EU IP DANSE approach. Paper presented at the meeting of the AiSoS (2013)
7. Paulen, R., Engell, S.: DYMASOS – Dynamic Management of Physically Coupled Systems of Systems. Published on ERCIM News 97 (April 2014); Special theme: Cyber-Physical Systems (February 25, 2014)
8. Coleman, J.W., Malmos, A.K., Larsen, P.G., Peleska, J., Hains, R., Andrews, Z., Payne, R., Foster, S., Miyazawa, A., Bertolini, C., Didier, A.: COMPASS Tool Vision for a System of Systems Collaborative Development Environment. In: Proceedings of the 7th International Conference on System of System Engineering, IEEE SoSE. IEEE (July 2012)
9. Siemieniuch, C., Sinclair, M., Lim, S.L., Henson, M.S., Jamshidi, M., DeLaurentis, D.: Project Title Trans-Atlantic Research and Education Agenda in Systems of Systems, T-AREA-SoS (2013)
10. CPS20: CPS 20 years from now - visions and challenges, a CPSWeek workshop, in Berlin, Germany (April 14, 2014), http://www.cyphers.eu/
11. Montecchi, L., Lollini, P., Bondavalli, A.: A DSL-Supported Workflow for the Automated Assembly of Large Performability Models. In: Procceedings of the 10th European Dependable Computing Conference (EDCC 2014), Newcastle upon Tyne, United Kingdom (2014)
12. Engell, S.: Cyber-physical Systems of Systems –Definition and core research and development areas. Working Paper of the Support Action CPSoS (2014), http://www.cpsos.eu/wp-content/uploads/2014/05/CPSoS-Press-Release_Project-Launch.pdf
13. Local4Global: SYSTEM-OF-SYSTEMS THAT ACT LOCALLY FOR OPTIMIZING GLOBALLY, Project Number: 61153, Project Start Date (October 01, 2013), http://local4global-fp7.eu/
14. Hodges, C.: COBWEB – Citizen Observatories Web: Ecology meets the crowd. In: GIS and Remote Sensing: the End of Fieldwork? Remote sensing and its role in ecological assessment, Chartered Institute of Ecology and Environmental Management (CIEEM) Welsh Section Conference and AGM, Aberystwyth (February 21, 2014)

15. Uhlir, P.F., Chen, R.S., Gabrynowicz, J.I., Janssen, K.: Toward Implementation of the Global Earth Observation System of Systems Data Sharing Principles. J. Space L. 35, 201 (2009)
16. Laudy, C., Petersson, H., Sandkuhl, K.: Architecture of knowledge fusion within an Integrated Mobile Security Kit. In: 2010 13th Conference on Information Fusion (FUSION), pp. 1–8. IEEE (July 2010)
17. http://www.shared-e-fleet.com/, http://www.iao.fraunhofer.de/images/iao-news/iao-news_june_2013_en.pdf
18. Pape, L., Agarwal, S., Giammarco, K., Dagli, C.: Fuzzy Optimization of Acknowledged System of Systems Meta-architectures for Agent based Modeling of Development. Procedia Computer Science 28, 404–411 (2014)
19. Wang, R., Dagli, C.H.: Computational System Architecture Development Using a Holistic Modeling Approach. Complex Adaptive Systems, 13–20 (2012)
20. Agarwal, S., Pape, L., Ergin, N., Dagli, C.: Multi-agent Based Architecture for Acknowledged System of Systems. Procedia Computer Science 28, 1–10 (2014)
21. Acheson, P., Dagli, C., Kilicay-Ergin, N.: Fuzzy Decision Analysis in Negotiation between the System of Systems Agent and the System Agent in an Agent-Based Model. arXiv preprint arXiv:1402.0029 (2014)
22. Agarwal, S., Dagli, H.C.: Adaptive Negotiating Strategies of Systems of Systems Coordinator using Deep Belief Networks. In: 4th International Engineering Systems Symposium, CESUN 2014 (2014)
23. Yang, X.S., Deb, S.: Multiobjective cuckoo search for design optimization. Computers & Operations Research 40(6), 1616–1624 (2013)
24. Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley & Sons (2010)
25. Wang, R.: Search-Based System Architecture Development Using a Holistic Modeling Approach. Ph.D dissertation, Missouri University of Science and technology (2012)
26. Wang, R., Dagli, C.H.: Developing a holistic modeling approach for search-based system architecting. Procedia Computer Science 16, 206–215 (2013)
27. Riddle, S.: Contract-based modelling and analysis technologies for Systems-of-Systems. In: 2012 7th International Conference on System of Systems Engineering (SoSE), July 16-19, pp. 469–470 (2012)

User-Centered Design for Emotion. A Case Study in Wellness Products

Sergio Gago Masagué and Joaquim Lloveras Macià

Abstract. The aim of the present work is to identify the attributes of the wellness products that affect users' experience. For this goal, we considered the values related to users' emotions, caused when the product stimulated their sensitivity. The study focused on three leading Spanish companies in the wellness industry. This study's most important tasks are: analyze current design processes and characteristics of each product, and perform an extensive data collection about users' satisfaction. The results of this research set new design considerations for wellness products on ergonomics, user safety and environmental awareness. We assigned a level of importance to each product attribute according users' feedback. The tabulation of these results created guidelines, that designers are using to recognize which product attributes most affect the user's experience as well as the magnitude of affect.

1 Introduction

This paper is focused on studying domestic wellness products from the point of view of user-centered design and emotional interaction. The interaction between these kinds of products and users and their capacity to awaken users' emotions is particularly relevant to the final design goal of providing relaxation and pleasure by stimulating several human senses through water (temperature and pressure), light, fragrances and music. Nevertheless, it has been noticed that most common

Sergio Gago Masagué
Department of Mechanical and Aerospace Engineering,
University of California, Irvine - 92667 Irvine, USA
e-mail: sgagomas@uci.edu

Joaquim Lloveras Macià
Departament de Projectes a l'Enginyeria, Universitat Politècnica de Catalunya,
Barcelona, Spain
e-mail: j.lloveras@upc.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_14

design tools target functionality and profitability, instead of user satisfaction. Most wellness companies in Spain do not have an effective mode of communication between the design team and the customer spas to integrate suitable emotional design features [1]. Due to preliminary users' feedback, we believe that more research is needed to apply current tools and provide new strategies for a better user-centered design in domestic wellness products.

Previous studies stated that the main factors affecting users' emotions are the human-product interactions and the user's perception [2,3]. When a product matches the user's aims and expectations, the user is satisfied. Users feel affinity for those objects that appeal to them and/or give them pleasure. Aesthetically or physically pleasing objects appear to the user to be more effective due to the creation of an emotional link with the object [4,5]. Thus, understanding how user emotions are related to products is especially important for designers, as it plays a crucial role in their success. It is important to note that not taking into account these emotional responses can be a missed opportunity for design optimization and increasing the market share [6].

2 Research Methodology and Data Collection

2.1 Aims of the Study

The main goal of our research is to review the design methodology implemented in three leading Spanish wellness companies while determining the emotional design deficiencies of two types of wellness products, the home spa and shower cabin. This determination is based on the internal data of three stakeholder companies and users' feedback collected through web-based survey. The design of this survey was specifically created for the present research. Preliminary data has already been published [1]. The final goal of this two year project has been to provide new tools in collaboration with the design teams of the companies to improve industrial wellness design.

2.2 Case Study

We have selected two domestic wellness devices for evaluation under an emotional design criteria: the home spa and the hydromassage shower cabin. They are both leading products in the sector, and both are the most complex, since they offer a wider functionality than any other product of this nature. In terms of features, the ones that should be highlighted are: hot tub, aromatherapy and / or chromotherapy, background music and waterfalls. Therefore, both products appear to be appropriate for the present case study, given that they have a large impact on users' emotional state by stimulating several senses at a time [7].

2.3 Sensory Attributes

Wellness products are composed of several elements. The elements that a user is able to perceive through his senses define the set of sensory attributes for each product, and these attributes may be perceived by users in two ways:

- Objective way (physical): This designates precise physical attributes that do not require a high level of interpretation. For example the color, texture, noise, etc...
- Subjective way (psychological): this designates all perceptions or feelings that are related to social, cultural and personal experience from the past. These intrinsically have a deeper value for the user.

The psychological perception of the user is determined by those human characteristics that, satisfied or not by sensory attributes, still affect a user's emotional tie to the product. These sensory characteristics should be a kind of language that communicates with each user, trying to transmit the information and feelings that the user needs at a given moment. If we could quantify the effect of these attributes, it would be possible to design products that could communicate the feeling or the particular effect desired by the user. However, this effect can only be offered through the physical stimulation of the user's sensory attributes: tactile, visual, olfactory and auditory senses. This is the reason it is important to analyze the specific sensory characteristics of the product. We classify the sensory elements attributable to a wellness product in Figure 1. Furthermore, Table 1 illustrates these attributes with examples of pleasant emotions:

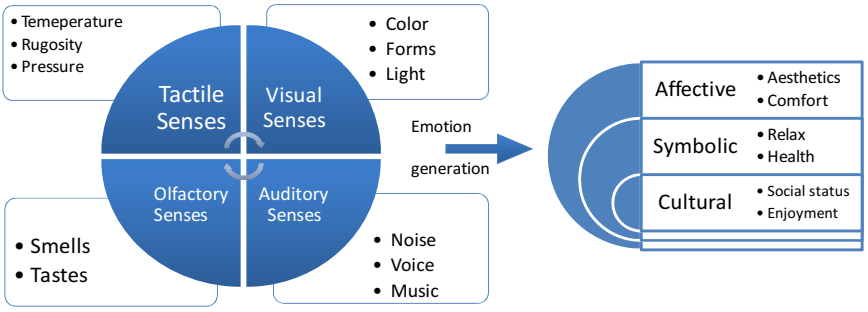


Fig. 1 Users' physical perception and emotion generation on wellness (Source: Authors)

Table 1 Type of pleasantness and examples of emotions

Type of pleasantness	Example of happiness/joy
Visual pleasantness	A beautiful external form or color
Auditory pleasantness	Silence and calm song while using
Olfactory pleasantness	Use of aromatic fragrance or soap in
Tactile pleasantness	Pleasant bubbly feeling of an air jet

2.4 Data Collection

In the framework of this research, an important data collection task has been developed. Two complementary sources of data have been used: on the one hand, existing internal data of three stakeholder companies collaborating in the project, and, on the other hand, online semi-structured surveys on the features of the products studied that were answered by some of the current users. A set of questions was designed to gather feedback from users. Questions tackled the following topics:

- The overall satisfaction with the product and with each of its functions.
- The relevance of these functions for the user.
- The main reasons for acquisition.
- The effect caused by product attributes.
- A final open field for suggestions to improve the product.

The interviewees who have been contacted for this purpose are clients of the most prominent Spanish retail companies in the Spanish wellness industry. A total of 327 users who bought a product have properly answered our online survey during 2012 and 2013, and the data was linked to the product sold. About 20% of the respondents are users of shower cabins, 50% of home spas and 30% own both products.

2.5 Methodology for Data Analysis

The results of the survey allowed us to identify the most relevant features for users of both the spa and the shower cabin, as well as to obtain users' satisfaction grades for these same attributes. We asked users to rate the importance of several features that one can find in the products we are studying. To simplify the results, we arrange the features rated as most relevant in group of attributes, as shown in Table 2. For example, the attribute "ecological" contains features such as water savings, energy efficiency and the disassembly and recycling program. We also asked about how satisfied users were with the features included in the product that they owned. In that way, we could rate the margin of improvement in the design for each of the rated attributes according to users. It is important to note that the

Table 2 Spas and shower cabins attributes. Margin of improvement

Product Attribute	**Importance**	σ_{Imp}	**Satisfaction**	σ_{Sat}	**Difference**
Effective hydromassage	8.75	1.02	7.00	2.15	+1.75
Ecological	8.25	1.16	4.00	2.39	+4.25
Intuitive	8.25	0.77	5.50	2.50	+2.75
Good aesthetics	4.75	2.39	8.00	1.84	-3.25
Comfortable / ergonomic / accessible	6.75	1.77	6.00	2.05	+0.75
Value for money	6.00	2.27	7.50	0.54	-1.50

* Grading continuous scale: 0 poor - 10 excellent

attribute "value for money" is not directly related with any feature that can affect user's interaction. However, this information will allow us to evaluate the impact in user's satisfaction of the economic investment to improve other features.

Figure 2 illustrates the data shown in table 2 as importance versus satisfaction. The elliptical bubbles represent the combination of the two sigma values; importance and satisfaction. The efficient design line represents the balance of these both attributes.

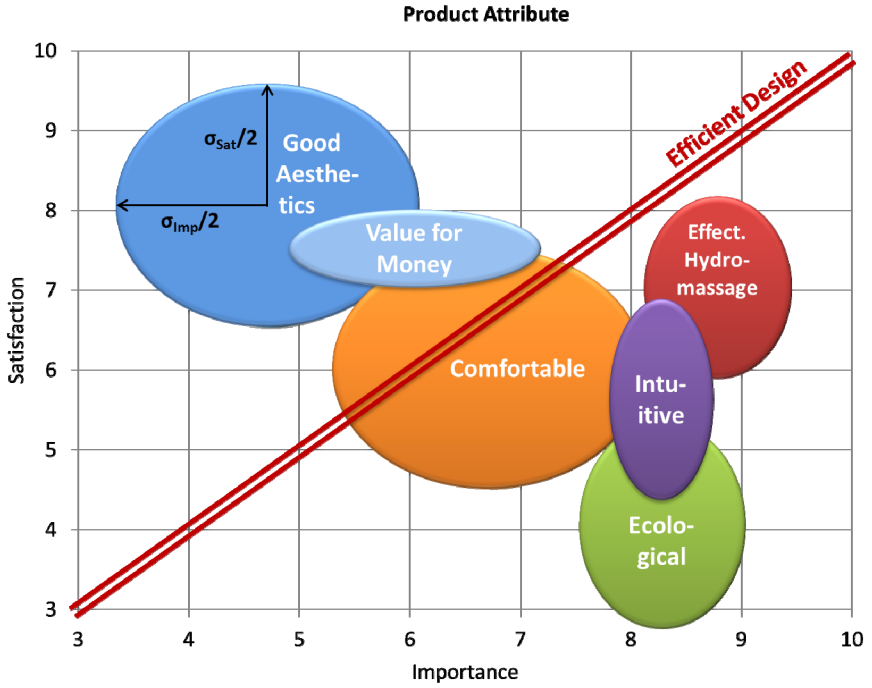


Fig. 2 Spas and shower cabins attributes. Importance versus Satisfaction (Source: Authors)

As we can see, the lowest rated attribute is the one related to ecological suitability, but that is also considered one of the most important. Therefore, this is the attribute in which designers may have a greater margin of improvement. The second-lowest rated attribute, intuitive, is connected to the simplicity of use and control of the product. Again, that attribute is contemplated by users as one of the primary requirements. As a consequence, substantial design improvement can be made in this regard. We can note also that the hydromassage function is defined as the most important and one of the most appreciated by users. Nevertheless, as this is the essential function of these products, we believe that it should be deeply analyzed for improvement, as it plays a crucial role in the users' evaluation. Finally, the previous results show that both the quality-cost relationship and the product aesthetic are overrated. This may be a result of great significance, as that implies important changes in the traditionally assumed design criteria for the spa

and shower cabin. This analysis may indicate that some of the attention paid to profitability and aesthetics can be diverted into creating eco-friendly and user-friendly wellness equipment for this particular case study.

The study of these attributes allowed us to detect major points of action. However, we needed to go beyond these attributes to analyze and identify the features and, eventually, the physical characteristics involved in each attribute. We list in Table 3 the main physical characteristics of these wellness products involved in users' perception, and we then relate these characteristics with product attributes. For example the physical characteristic of water temperature may affect the value of attributes such as effectiveness, comfort and enjoyment. Characteristics such as aromatherapy, chromotherapy and musicotherapy may affect the value for money, as they are included in high-end products.

Table 3 Physical characteristics involved in the product attributes

Physical Characteristics	Product Attributes
Water temperature	Effectiveness, comfortable, hydrotherapy, enjoyment
Water pressure	Effectiveness, comfortable, hydrotherapy, enjoyment
Roughness of materials	Effectiveness, comfortable, accessible
Smells / tastes	Effectiveness, comfortable, enjoyment, value for money
Color / shape	Intuitive, comfortable, ergonomic, accessible
Lighting system	Effectiveness, comfortable, intuitive, enjoyment, value for money
Noise	Comfortable, intuitive, value for money
Music	Effectiveness, comfortable, intuitive, enjoyment, value for money

Using the detailed survey data, these final values will be weighted depending on the type and model of product. Therefore, the appropriate parameters should be chosen to compare these values with the functions that are impeding the user's satisfaction. The ratio of sensitive parameters to improve the product functions requires a deep understanding of the product, such us features offered, how they are designed, functional limitations and range. Obtaining new points of action to improve the user experience requires, not only a precise study of the data regarding user feedback, but also interpretation of the values with the design team to identify areas for improvement and economic and technological feasibility. We started on the level of satisfaction reported by users for the product sold. Then, we went through the attributes involved to find the particular physical characteristic causing a poor user satisfaction rating. After determining the weak point, we

proposed several solutions and discussed their feasibility with the design team. In the example shown in Figure 2, we studied user satisfaction in ergonomics and comfort. Following this new methodology, we detected chances of improvement in the location and size of the buttons of the control interface. More details about the control interface study can be found in Section 3.3.

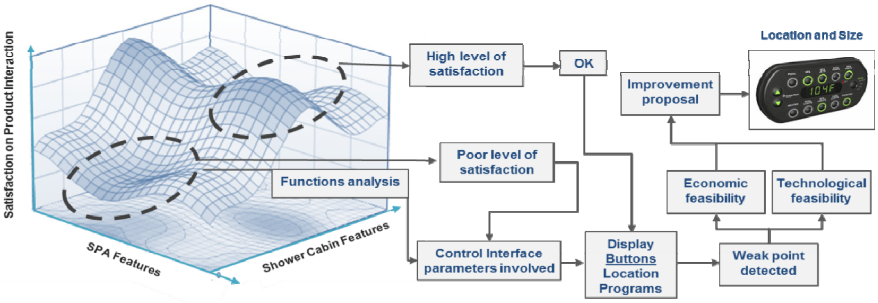


Fig. 3 Workflow: users' satisfaction, product feature, and design parameters (Source: Authors)

3 Weak Design Points Detected

Specific questions about the features provided us enough information to identify points of action and propose new consideration for design. Among the features analyzed, the most important reported by users are those in the hydromassage function, the frame of the spa, the control interface and the environmental awareness.

3.1 Hydromassage Function

It can be inferred that the hydromassage is considered by users (72% of total) as one of the most important functions of both the spa and the shower cabin. Features of the water jet, pressure, thermal sensation and noise are the weak points identified to improve this value.

Customizing Alignment and Pressure of the Water Jets. The final goal of hydromassage is to soothe the user's points of discomfort. Thus, adjusting the location and alignment of the jets to meet the human body trigger points is a key design factor. In particular, the position, alignment and pressure of the water jets appear to have a great impact on users' overall satisfaction. More than 15% of total users reported discomfort regarding the current configuration on alignment and pressure. Weak or incorrect regulation of these variables results in a poor assessment of the product. Consequently, a design proposal to improve this weak point could allow users to easily regulate both the pressure and the slope of the water jets.

Water Splashing. Water splashing into the eyes and airways is considered by 12% of users to be an annoyance when using spas, especially if the water contains salt or chemicals. One major change reducing this effect could consist of regulating the intensity of the upper jets of the spa, avoiding excessive bubbling in areas close to the user's face. The control of pressure for these jets should be independent from the rest.

Thermal Sensation. Thermal comfort is an aspect of great importance to users' experience. Users' first contact with water appears to be unpleasant in 17% respondents (11% from showers and 6% from spas). In the case of the shower cabin, the main problem reported was that the temperature of the water that remains in the pumps and pipes after the last use cannot be controlled, and what usually results is a first stream of cold water in a later use of the device. For the spa, negative reviews about the thermal sensation during the first immersion have also been reported. Some users find it too cold, while others state just the contrary. Given these conflicting answers, we could propose a new product feature allowing users to customize the initial temperature of the water.

Ambient Noise. The answers showed that the vibration of water and air pumps causes an unpleasant noise that disturbs the relaxation of 8% of the total users involved in the study. This noise has a much greater repercussion on the spa (6% of users). We can explain this fact noting that the falling water in the shower cabin produces a sound which is louder than the one in the pumps. A proper solution may be to improve the location and the acoustic insulation of the pumps inside the structure of the spa.

3.2 Frame of the SPA

The data shows that more than 20% of the users have issues with characteristics regarding the frame of the spa. Among these features, we can locate features involving ergonomics and safety.

Ergonomics. Almost 17% of the users reported a low rate of comfort in the spa in terms of ergonomics. The issue was reviewed by several designers who concluded that that current design trends bet on straight lines. These aesthetics could be adopted for the external design, but not for the interior, as they do not fit the body shape, resulting in an uncomfortable place to sit or lie down for a while. We believe that a clear distinction has to be made between the criteria to be applied in the design of the outer casing and the criteria used in the interior design.

Undesired displacements. The floating effect in spas, especially noticeable in devices using salt water, appears to be a reason for discontentment in users. When floating, the user is horizontally displaced by water currents, as water jets push him from one side to the other. In order to reduce these undesired displacements, a redesign of the existing headrest cushion is suggested. Apart from providing support, these pads should also work as fixation elements.

Safety Features. 7% of the users involved in the present research reported to be worried about suffering an accident when entering or exiting their spa. We must

note that there were about 230,000 bathroom injuries in just the United States during 2008, and 37% of these injuries took place when bathing, showering, or getting out of the tub or shower [8]. These accidents are especially frequent for elderly users who have been exposed to a prolonged use. Research on the field has shown that a large number of spas (74% sold by the companies involved) do not provide a safe access or egress. Consequently, new proposals should address this problem, such as improved anti-slip surfaces and support handles.

3.3 Control Interface

With respect to the control interface, products exhibit poor quality to users in terms of either visibility or interpretability of commands (7% of users) and accessibility of the controls (9% of users). Only very high-end spas and shower cabins feature a tactile interface with clear and total control over the devices' functionality. In addition, most users do not believe the design and/or the location of the control interface is intuitive and convenient (16% of users). Regarding the data obtained, the proposals to address these problems could be, first, to design both buttons and/or touchscreens large enough in order to make their use easy and avoid recurrent typing errors. Alternately, the controls could be located in a more reachable position for the user.

3.4 Environmental Awareness

The results show that consciousness also influences users' interaction, up to the point that 49% of the users communicated that they did not fully enjoy a prolonged or frequent use of the device if that meant wasting water. Therefore, environmentally friendly strategies, such as using purification or recycling water systems, should be promoted to avoid this bad feeling. This aspect is more emphatic in the case of the shower cabin, since in the spa, which is a kind of pool, the water can be physically and chemically treated to make it suitable for reuse. The existing shower cabins do not have the capability of recycling the water. In this case, the redesign proposal would consist of implementing a reflow water circuit.

4 Generalizing Results to Improve Current and Future Designs

Each of the weak points identified in the previous section has been deeply studied in teams and assigned several design considerations to be checked in order to guarantee a more pleasant user experience. They have been classified in terms of the human perceptions that are concerned. These results have been summarized and presented into tables in this section, making them easier to interpret.

4.1 Design Guidelines and Points of Control

Hydromassage Function Guidelines. The spa is a more relaxation-oriented product than the shower cabin. This fact could explain why spa users are more sensitive to high noise levels exposure than shower cabin users. The most relevant attributes of the spa related to its hydromassage function are: the position, alignment and pressure of water jets; the noise levels generated by the pumps; the initial temperature of the water (users' thermal sensation) and the temperature during use. The elements that need to be improved in order to reduce the negative impacts on the user are: the jets, the pumps and the control software application. The qualities that these three elements should present in order to fulfill the users' requirements are the following. For the jets, the pressure and the alignment should be adjustable, and the location of the jets should meet the position of the human body trigger points. The pumps, the combination of location, shape and internal operation factors should give a lower noise level outcome. For the software application, should provide the regulation of both the initial temperature of the water (before first immersion) and regulate temperature during use. The pressure and alignment of water jets should also be automatically adjustable.

Spa Frame Guidelines. The considerations that the designer should infer from the guidelines regarding the frame of the spa mostly involve ergonomics and safety elements. The interior needs to be as ergonomic as possible. Safety elements such as anti-slip surfaces and support handles are important both on the inside and the outside of the equipment. The cushion should be designed in order to provide not only support but also to fix the user's body in place and thus avoid undesired flotation and displacement.

Control Interface Guidelines. The design considerations for the control interface relate primarily to its location and the characteristics of the buttons. The location of the control panel should be easily and effortlessly accessible. The operation of the control buttons should be softer, as it was reported to be too hard. The keyboard and/or the touchscreen should be big enough in order to make for easy use and to avoid recurrent typing errors.

4.2 Summary of Guidelines

All the previous design aspects have been gathered into Table 4. This table can be used as a tool to assess the fulfillment of the emotional design criteria that are applicable to the current state of the art of the spa and the shower cabin.

Table 5 contains information about the grade of importance that users allocated to each of the evaluated features. Using the survey data shown in Table 2, it is possible to assign a grade of users' importance to each of the product attributes studied. This therefore gives the designer ideas about the relative impact that design improvements will have on user satisfaction. Thus, this table can be used to prioritize interventions detailed in Table 4.

Table 4 Summary table of guidelines for the validation of sensory attributes

<table>
<tr><th colspan="2" rowspan="3">Product / Part</th><th colspan="7">User' sense / Product attribute</th></tr>
<tr><th colspan="3">Tactile Sense</th><th colspan="3">Visual Sense</th><th>Auditory Sense</th></tr>
<tr><th>Temp.</th><th>Pressure</th><th>Rough.</th><th>Size</th><th>Shape</th><th>Position</th><th>Noise</th></tr>
<tr><td rowspan="1">CABIN</td><td>Water-flow Circuit</td><td>Contact feeling</td><td>Customi-zable</td><td colspan="2">free</td><td colspan="2">Allow used-water reflow</td><td>(<20dB)</td></tr>
<tr><td rowspan="8">SPA&CABIN</td><td>Player support</td><td colspan="3" rowspan="3">n/a</td><td colspan="3">Protects against water and humidity (sealed)</td><td rowspan="6">n/a</td></tr>
<tr><td>Led location</td><td colspan="2">free</td><td>Indirect</td></tr>
<tr><td>Interface - Frame</td><td colspan="2">free</td><td colspan="2" rowspan="2">Centralized and accessible</td></tr>
<tr><td>Interface - Buttons</td><td rowspan="2">n/a</td><td>Hardness Control.</td><td>not slide</td><td>not induce typos</td></tr>
<tr><td>Interface - Display</td><td>n/a</td><td rowspan="3">n/a</td><td colspan="3">Can be read from user' location</td></tr>
<tr><td>Jets</td><td rowspan="2">free</td><td rowspan="2">Customi-zable</td><td colspan="3">Adaptable trigger points</td></tr>
<tr><td>Water/air pumps</td><td>free</td><td>free</td><td colspan="2">noiseless (< 20dB)</td></tr>
<tr><td rowspan="3">SPA</td><td>Inside</td><td>1st thermal sensation</td><td rowspan="3">n/a</td><td rowspan="3">Non-sliding</td><td colspan="3">Adapted to human's body shape</td><td rowspan="3">n/a</td></tr>
<tr><td>Outside</td><td rowspan="2">free</td><td colspan="2">Supports for safe access</td><td>free</td></tr>
<tr><td>Pillow</td><td colspan="3">Allows user to fix position</td></tr>
</table>

Table 5 Summary table of user's valuation of sensory attributes

Product / Part		User' sense / Product attribute						
		Tactile Sense			Visual Sense			Auditory Sense
		Temp.	Pressure	Rough.	Size	Shape	Position	Noise
CABIN	Water-flow Circuit	A				A		C
SPA & CABIN	Player support				C			
	Led location						C	
	Interface - Frame						B	
	Interface - Buttons		B		B			
	Interface - Display		B				B	
	Jets		A				A	
	Water/air pumps		A					C
SPA	Inside	A					C	
	Outside			C		C		
	Pillow						A	

* Grading scale: C: Important – B: Very important – A: Most important

4.3 Integrating Guidelines in the Design Process

Based on both the preferences of users and the aforementioned detected weaknesses of the analyzed wellness products, we have established a set of "emotional design-based" guidelines to be implemented in the current design process of Spanish companies [9,10]. We propose integrating these guidelines in the validation stages, and they should be used for evaluating the main sensory features and functions of these products. In Figure 3, the proposed validation guidelines (called "sensorial control") are integrated in the overall design process, which is outlined by means of a flowchart of three basic stages: strategy design, concept design and detail design. The proposed guidelines should help to validate the concept design for each product part listed in Table 4. In particular, attributes such as size and position can be important at this stage. We propose validation of the design of other features such as shape, pressure, roughosity and noise using Tables 4 and 5 during the stage of detail design. In addition, the tools provided can help designers to gather users' feedback to better apply other well-known emotional techniques in the design for ergonomics and human factors, such as Kansei techniques [11].

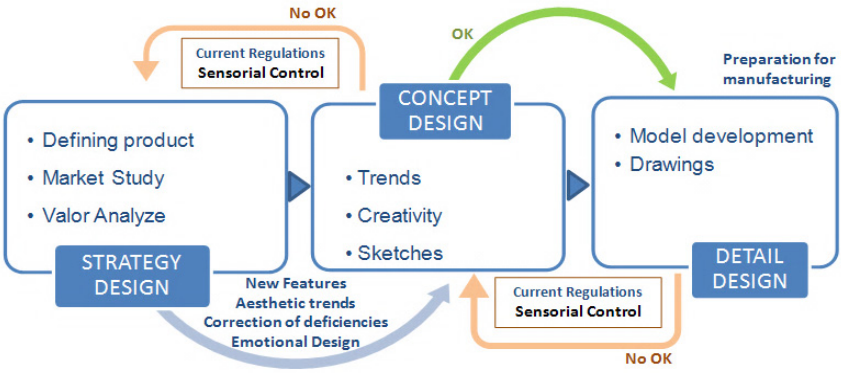


Fig. 4 Integrating Guidelines in the Current Wellness Design Process (Source: Authors)

5 Results and Discussion

Among the features analyzed, the most important that users have reported are those involved in the hydromassage function, the frame of the spa, the control interface and environmental awareness. Linking these global features with particular characteristics of the product has resulted in a set of actions points to consider. In the present study, weak design points regarding users' satisfaction have been identified thanks to specific questions about the features involved in the product-user interaction. Moreover, the affect of these design features on user experience has been graded. This is especially useful for redesigning current products, since it allows estimating which design improvements will have the greatest impact on user experience. In other words, it is possible to determine the most profitable investment for an emotional design improvement. These validation guidelines have already been tested by one of the stakeholder companies to evaluate the sensory qualities of one spa and one shower cabin. We envision a more successful design for these products reflected in a higher market share.

Now that the study has been concluded, we can state that the analysis of the current methodology of wellness design highlights the limitations suffered by the companies in finding attributes to provide added value to their products in terms of users' pleasantness. Additionally the intent by the companies studied of applying techniques to design for emotion, such as Kansei engineering, was not successful. Poor communication between current users and designers seems to be one of the main problems in finding reliable bases for emotional design improvements. It is important to note that the considerations resulting from the present study are not intended to limit designers' freedom. They are not meant to preset any specific feature of the product, as they depend on the product range. Nevertheless, this study can be considered a basis for using new technologies to gather users' feedback for wellness devices, and applying this information to the improvement of current design methodologies.

We also want to note that we surveyed current users to grade the product's attributes in terms of importance and satisfaction. This is the first approach to identify the main weak points of design. Future work in this research may include an in-depth detailed study of the users' emotion triggered by each feature. A good strategy would be to describe and measure emotional states using the well-known PAD emotional state model. This model is based on 3 axes: P-axis (pleasure), A-axis (Arousal) and D-axis (Dominance) which would allow researchers to better understand how each feature contributes to the user's emotion [12].

To enhance the external validity of this study, users and companies from different countries would need to be analyzed. In addition, the conclusions on cost and quality, as well as product aesthetics, are based on a population that already owns this kind of product, which is considered a luxury product. Therefore, it would be difficult to apply the cost value found to design any other group of wellness products, as the importance of the price for other kinds of product may have a higher value.

References

1. Gago, S., Lloveras, J.: Human Emotional Interaction and Hydrotherpy: Industrial Design Keys. In: Aiguier, M., Caseau, Y., Krob, D., Rauzy, A. (eds.) Complex Systems Design & Management, vol. 126, pp. 259–272. Springer, Heidelberg (2013)
2. Heller, D.: Aesthetics and Interaction Design: Some Preliminary Thoughts. Interactions 12(5), 48–50 (2005)
3. Norman, D.A.: Emotional design: Why we love (or hate) everyday things, 2nd edn. Basic Books, New York (2005)
4. Chapman, J.: Design for empathy: Emotionally durable objects and experiences. VA Earthscan, Sterling (2005)
5. Koskinen, I., Battarbee, K.: Empathic Design. User Experience in Product Design, pp. 119–130. Helsinki IT Press (2003)
6. Norman, D.A.: Emotion and design: Attractive things work better. Interactions ix(4), 36–42 (2002)
7. ISPA, ITEC, Los consumidores de spa en el mundo, Paris, ITEC (2011)
8. Stevens, J.A., Haas, E.N., Haileyesus, T.: Nonfatal bathroom injuries among persons aged ≥15 years–United States, 2008. Journal of Safety Research 42(4), 311–319 (2011)
9. Pugh, S.: Total Design: Integrated Methods for Successful Product Engineering. Addison-Wesley, Wokingham (1991)
10. VDI 2221, Systematic Approach to the Design of Technical Systems and Product. VDI-Verlag GmbH, Dusseldorf (1987)
11. Mitsuo, N.: Kansei Engineering. In: Stanton, N., Hedge, A., et al. (eds.) Handbook of Human Factors and Ergonomics Methods, pp. 83.1–83-5. CRC Press LLC, USA (2004)
12. Bales, R.F.: Social interaction systems: theory and measurement, pp. 139–140 (2001)

A Fuzzy Approach for Assessing Transportation Infrastructure Security

Michelle S. Dojutrek, Samuel Labi, and J. Eric Dietz

Abstract. The security of any transportation infrastructure can be defined as a combination of threat likelihood, infrastructure resilience, and consequence. In view of their inherently dynamic and highly unpredictable nature, threat likelihood and consequence data is difficult to determine with certainty. Due to this problem, this paper presents a new fuzzy methodology to qualitatively determine the overall security level, in terms of a security rating, for transportation infrastructure by duly considering the uncertainties of the environmental threats it faces, its resilience to damage, and the consequences of the infrastructure damage. The method is useful when data is unavailable or imprecise, allowing the security rating to be determined using a qualitative expert-assigned level that each factor contributes to overall security. The evaluation of the security factors are represented as fuzzy triangular numbers with accompanying membership rules that define the extent of contribution by each factor to overall infrastructure security. Through a case study, the paper applies the methodology to illustrate how general data can be used in the method to determine the overall security of specific infrastructure.

1 Introduction

Transportation comprises many modes of travel including air, rail, vehicle, waterway, and pipeline. Due to their typical large diversity, size, and connectedness, national transportation systems are vital to the economy and

Michelle S. Dojutrek · Samuel Labi
Purdue University, Department of Civil Engineering, West Lafayette, IN, USA

J. Eric Dietz
Purdue University, Department of Computer and Information Technology,
Purdue Homeland Security Institute, IN, USA
e-mail: {mdojutre,labi,dietz}@purdue.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_15

security. The abundance of infrastructure networks has led to their criticality in performing the functions of everyday life as well as their interconnectedness with other infrastructure networks, industries, and workforces that rely upon them (Barker et al., 2013). The transportation industry enables a monumental amount of passengers and goods to move throughout the world annually (Polzin 2012; DHS, 2011). Activities in the U. S. transportation sector make up 12% of the gross domestic economy and most businesses rely on a functioning transportation system to move their products. The U.S.'s Marine Transportation System, including ports, waterways, and vessels, handles more than $900 billion in international commerce every year (Lundquist, 2011). Freight revenue on U.S. railroads in 2010 was $56.3 billion with coal taking up 43% of the total commodities shipped (AAR, 2012). Airlines in the U.S. totaled $134.7 billion for 2011 revenue, 6.8% higher than the previous record set in 2008 (Herbst, 2012).

The failure of transportation infrastructure could occur as a result of any of multiple types of events. Unintended termination of transportation infrastructure is caused by infrastructure failure such as design flaws, fatigue, advanced deterioration and other internal causes (Labi, 2014). Infrastructure is also affected by unintended outside forces such as overloading, accidents, natural events and other external causes. Additionally, infrastructure may be damaged by intentional man-made forces such as terrorism. Transportation infrastructure makes attractive targets of intentional harmful attacks because of their visibility, accessibility, and capacity to carry large numbers of commuters in a relatively confined space (Steffey, 2008). Maritime and surface transportation systems are vulnerable to attacks by terrorists who seek to attract publicity, inflict high numbers of civilian casualties, and cause political and economic disruption (Harris et al, 2012). The range of potential threats to infrastructure is wide and if the infrastructure can withstand these effects by being bolstered against likely threats, consequences can be reduced. Due to the need for openness and accessibility at thousands of entry points, the wide geographic distribution of infrastructure, and the static nature of some routes, complete protection of surface transportation infrastructure is simply not feasible (Steffey, 2008). Therefore, creating infrastructure that is resilient would mitigate the need for continuous efforts to oversee the security of infrastructure with limited manpower.

Ezell et al, 2000 framed five key steps to risk management to provide evidence for security investments to subsequently reduce the overall infrastructure damage in the event of disaster:

- **Measure** the threat likelihood posed by external or intentional threats to the asset
- **Monitor** the threat likelihood over time
- **Assess** the effectiveness of actions intended to reduce consequences
- **Communicate** this information to the general public and legislators
- **Provide** evidence for appropriate resources

It has been argued that both the scale and nature of transportation systems necessitate that a reasonable degree of risk must be accepted, even for critical infrastructure, because complete mitigation is not feasible. Formal methodologies for assessing and managing risks to transportation security provide a valuable conceptual structure and practical tools for allocating resources in cost-effective ways to improve public safety (Steffey, 2008). The United Kingdom policy, Publicly Available Specification 55 Part 2 (PAS 55-2), states that risk management is fundamental for proactive infrastructure management and that its purpose is to understand the cause, effect and likelihood of adverse events occurring in order to manage these risks to an acceptable level; Also, the International Infrastructure Management Manual (IIMM), a guidance document focused on experience in Australia, New Zealand, UK, South Africa and the US (INGENIUM, 2006), recommends that core infrastructure management should identify critical infrastructure and events and apply risk management to these infrastructure (Hooper et al., 2009).

The uncertain nature of security factors such as threat likelihood, infrastructure resilience, and consequence must be considered in any security metric (Dojutrek et al, 2014). For example, hazards are highly non-deterministic such as the magnitude of earthquakes or the number of accidents, and cannot be predicted at 100% accuracy. The failure to consider uncertainty could lead to infrastructure being unprepared for the potential range of hazards that will act on the infrastructure and therefore cause greater consequences (Dojutrek et al, 2014). Additionally, infrastructure that is highly resilient to hazards in the area could potentially withstand the threat and therefore cause little resulting consequences. Uncertainty can be accounted for by using historical data trends, predictive models, and expert opinion to provide a range of threat likelihood, consequence, and resilience scenarios that would potentially affect the infrastructure. Unexpected damage from natural occurrences and man-made incidents increase infrastructure repair costs and lives lost. If infrastructure is made resilient against these threats, damage and costs can be reduced. Thus, a metric for identifying infrastructure in need of improvements for security purposes can help prioritize infrastructure for the limited funds dedicated to transportation needs. To capture the dynamic nature of the security factors, fuzziness will be used in the development of the security metric. Furthermore, due to the uncertain nature of threats (their occurrence and magnitudes cannot be predicted with complete certainty (Dojutrek, 2014)), it is vital to incorporate concepts of uncertainty in any analysis that deals with risk prediction and security investment evaluation. Failure to consider uncertainty can lead to overestimation or underestimation of the likelihood of the threat, damage to the infrastructure, and consequences of the damage to the community. Uncertainty can be quantified by analyzing historical data trends and developing models for threat likelihoods and magnitudes, infrastructure damage due to the threat, resilience enhancement due to the security investments, and community consequences of threat occurrence.

2 A Review of Past Work

The Oxford English Dictionary (online) defines risk as "(Exposure to) the possibility of loss, injury, or other adverse or unwelcome circumstance; a chance or situation involving such a possibility." Different definitions of risk exist, but in relation to infrastructure management, a risk definition should involve the combination of probability and consequence of any uncertain event (Hooper et al., 2009). Quantifying and assessing risk involves the calculation and comparison of probabilities, but most expressions involve compound measures that consider both the probability of harm and its severity; thus, quantitative risk assessment is an important growing component of the larger field of risk assessment that includes priority setting and management of risk (Melnick and Everitt, 2008).

Due to the difficulties in quantifying key components of threat, vulnerability, and consequence assessments, analyses of transportation security risk typically employ qualitative methods in making judgments about the relative magnitudes of various risk scenarios (Steffey, 2008). There is concern that in the allocation of general resources or security-specific funding, only high valued infrastructure would receive high priority. A weighted qualitative approach could be used to ensure that the lower-valued infrastructure receive due consideration during the evaluation process (Dojutrek et al, 2014). For example, a specific measure involving asset size or asset cost could be weighted higher or lower to allow other factors to gain more importance in the framework.

Threat, resilience, and consequence information are involved in risk assessment while risk management involves deciding which protective measures to take based on an agreed upon risk reduction strategy (Moteff, 2005). Risk is a multidimensional concept which is often expressed as the Cartesian product in the context of risk analysis for critical infrastructure (McGill et al., 2008):

$$Risk = threat \bullet infrastructure\ vulnerability\ or\ resilience \bullet consequence \quad (1)$$

Where, *threat* is an adverse event, *consequence* is the repercussions of the infrastructure loss, and *infrastructure vulnerability or resilience* is the target weaknesses that can be exploited by an adversary to achieve a given degree of loss or cause the infrastructure to fail during a natural hazard.

The TMC Risk Assessment Methodology (TCM RAM) is a combination from three different sources, the Systematic Assessment of Facility Risk (SAFR), a methodology developed by the DHS Office of Domestic Preparedness in its toolkit, and ideas from AASHTO's Guide for Vulnerability Assessment (SAIC, 2005). The steps in this method include: infrastructure identification, threat assessment, consequence assessment, vulnerability assessment, and countermeasure development, and it evaluates risk in terms of a terrorist attack using Equation 2.

$$RR = TA \cdot T \cdot C \cdot (1 - LD) \cdot (1 - LS) \quad (2)$$

Where, *RR* (Relative Risk) is a function of the overall threat to the infrastructure or facility, *T*; the attractiveness of a particular target to a given adversary, *TA*; the potential consequences of a successful attack on a target, *C*; the ability to deter an adversary from attempting an attack, *LD* (expressed in terms of the inability to deter, or *1-LD*); and the effectiveness of the system to prevent an attack should one be attempted, *LS* (expressed in terms of system ineffectiveness, or *1-LS*).

This method evaluates vulnerability and criticality in terms of relative risk and target attractiveness. However, calculating the relative risk for specific infrastructure has limited value because it indicates only the risk associated with that infrastructure relative to the highest and lowest possible *RR* values (Venna and Fricker, 2009). The TMC RAM is a theoretically good model, but requires a lot of expert effort to quantify the value of subjective criteria which could introduce inconsistency and variance into the model.

Xia et al. (2004) developed a framework for risk assessment that includes static and dynamic infrastructure characteristics in the event of a terrorist attack. The risk score of a highway component is defined as a linear combination of three indices:

$$R = (\propto A + \beta B) \cdot \frac{C}{100} \tag{3}$$

Where, *R* is the risk score of highway network component; *A* is the static characteristic index; *B* is the dynamic characteristic index; *C* is the attack potential index; α is the weight of the static characteristic index; and β is the weight of the dynamic characteristic index.

The static characteristics (Index A) include: structural stability, number of alternatives, and response resources of highway components. The dynamic characteristics (Index B) include: dynamic traffic flow information such as volume, speed, occupancy, vehicle classification, and queue length as well as weather details and work zone activities. The potential of a terrorism attempt (Index C) is estimated in terms of functional significance and symbolic importance of a highway component (Xia et al, 2004). The study did not include uncertainty.

The score of Indices A, B, and C are calculated as:

$$A = aW_{A1} + bW_{A2} + cW_{A3} + dW_{A4} \tag{4}$$

$$B = eW_{B1} + fW_{B2} + gW_{B3} + hW_{B4} + iW_{B5} \tag{5}$$

$$C = jW_{C1} + kW_{C2} \tag{6}$$

Where, *W*'s are the weights predetermined with the help of experts; and *a, b, c, d, e, f, g, h, i, j, k* are characteristics pertaining to each index.

McGill and Ayyub (2008) used fuzzy logic to approximate the true functional relationship between the effectiveness of six security system capabilities (access control, personnel barriers, vehicle barriers, surveillance systems, guard force, and reaction force with heavy weapons) and probability of adversary success. The goal of the model is to provide a system based on approximate reasoning that produces an estimate for the probability of adversary success based on the subjective evaluation of several or more defensive criteria. $Pr(S|A_i)$ is the probability of adversary success (S) given the occurrence of initiation event A_i and the complementary event $Pr(\bar{S}|A_i)$ as the security system effectiveness. Each defensive criterion (six security system capabilities) can take on a linguistic value of "Low," "Medium," or "High" defined on a constructed scale for effectiveness with membership functions. The consequent Pr(S|A) may take on linguistic values such as "Likely," "Certain," or "Even Chance." There is the possibility that each defensive criterion may require its own set of linguistic phrases for effectiveness, for example if one criteriaon was based on a constructed scale and another on a crisp scale such as time. A user (security expert) can subjectively assign a value to each premise of criterion on a scale of 0-10 or an alternate scale for a given facility of infrastructure and attack type once the fuzzy inference rules are defined.

Another study by Yazdani et al., (2012) identified the risk criteria and used Fuzzy TOPSIS as an uncertainty-based multi-criteria decision-making technique to determine the weights of each criterion and the importance of investment alternatives with respect to the risk criteria. This framework extends the Risk Analysis and Management for Critical Infrastructure Protection (RAMCAP) method by introducing new parameters to assess the effects on risk value. According to the authors, the TOPSIS method helps decision-makers carry out analysis and comparisons in ranking their preference of the alternatives with vague or imprecise data. It is based on the concept that the chosen alternative should have the shortest distance from the most ideal solution and the farthest distance from the least ideal solution.

A study by Yang et al. (2009) uses a fuzzy evidential reasoning (ER) method to conduct maritime security assessments. The authors developed a subjective assessment and management framework using fuzzy ER approaches. The consequence parameter is a security parameter which can be derived from multiple risk parameters: will, damage capability, recovery difficulty, and damage probability. *Will* is the likelihood of a threat-based risk, which directly represents the lengths a malicious attacker goes through in taking a certain action. To estimate *will*, one may choose to use such linguistic terms such as "Very weak," "Weak," "Average," "Strong," and "Very strong." The combination of *damage capability* and *recovery difficulty* represents the consequence severity of the threat-based risk. Specifically speaking, *damage capability* indicates the destructive force/execution of a certain action and *recovery difficulty* hints at the resilience of the system after the occurrence of a failure or disaster (Yang et al., 2009). The following linguistic terms can be considered as a reference to be used in subjectively describing the two sister parameters: "Negligible," "Moderate,"

"Critical," and "Catastrophic" for *damage capability* and "Easy," "Average," "Difficult," and "Extremely Difficult" for *recovery difficulty*. *Damage probability* means the probability of the occurrence of consequences and can be defined as the probability that damage consequences happen given the occurrence of the event, and could be described using terms such as "Unlikely," "Average," "Likely," and "Definite" (Yang et al., 2007; Yang et al., 2009).

The TMC RAM methodology identified key aspects of infrastructure that terrorists may consider in an attempt to cause destruction and developed a risk equation to capture these factors, but did not consider dynamic concepts or uncertainty of the variables. The method developed by Xia et al, (2004) addressed the dynamic nature of specific infrastructure aspects without including uncertainty. McGill and Ayyub (2008) developed a fuzzy approach to assess the effectiveness of security system capabilities from the terrorist perspective, but did not look specifically at infrastructure characteristics or the natural threat perspective. Yazdani et al, (2012) added two new criteria into the traditional risk equation and input the new criteria into a fuzzy framework. Yang et al, (2009) further developed the variables used in the traditional risk equation to include new parameters based on terrorist attack for maritime transport and input these into a fuzzy evidential reasoning framework. The method does not break down the variables into infrastructure specific subcategories.

Based on limitations of past studies, the method described in the next section is further developed to include fuzzy logic to capture the dynamic and uncertain nature of each identified security factor for transportation infrastructure. The method further breaks down each factor into additional measures and attributes which are also fuzzified. This allows each infrastructure characteristic to be qualitatively assigned a level of influence based on expert opinion. The fuzzy output therefore provides decision makers with a method to capture the security level of an infrastructure without precise or detailed data; rather it allows experts to make decisions based on their experience by qualitatively assigning levels to each variable of security in the method.

3 Methodology

3.1 Security Rating

A synthesis of past work has generally shown that the security of an infrastructure is a function of three main factors: the Threat Likelihood, Infrastructure Resilience, and Consequence (Dojutrek et al, 2014). The combined effect of these three factors is a security rating metric. This paper duly accommodates the fact that all three factors are characterized by a significant degree of uncertainty and therefore introduces fuzziness in the levels of these factors and subsequently, in their outcome (i.e. security rating). The enhancements to the method with allow experts to use the security rating method without imprecise data. Figure 1 illustrates the framework used in the paper.

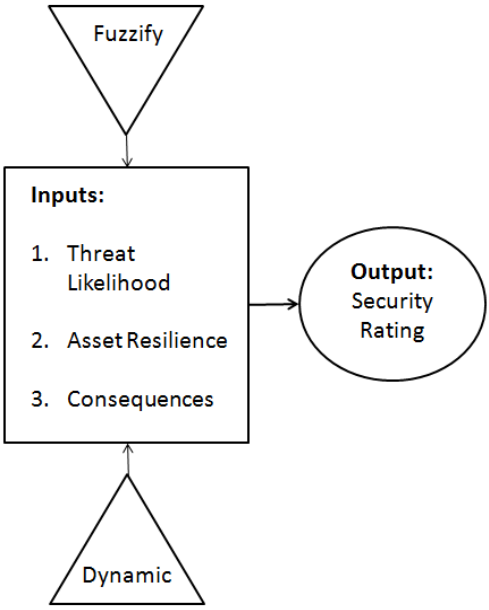


Fig. 1 Framework

The three inputs that influence infrastructure security are herein referred to as security factors. Each factor has a set of measures that quantify how much the factor contributes to overall infrastructure security. Each measure is further decomposed into a set of associated attributes that determine the level of the measure rated on a scale to define the overall amount that the measure contributes to the factor (Dojutrek, 2014). Most attributes have different units, therefore the attribute data were scaled to address these dimensional inconsistencies by rating them on a scale of 1 to 5, 1 representing an attribute level associated with high security and 5 representing an attribute level associated with low security.

The formulation of the security rating expression is shown in Figure 2. The overall security rating equation used to determine the level of infrastructure security is shown in Equation 7. For a high level of threat likelihood and consequence, and a low level of resilience, the security rating decreases. For a high level of resilience, and a low level of threat likelihood and consequence, the security rating increases. This metric is useful in determining if infrastructure is secure and how the infrastructure compares to others in the system.

Factor	$F_i = w_1 M_1 + \cdots + w_j M_j$ Where F_i is security factor i, w_j is weight of measure j, and M_j is security measure j
Measure	$M_j = \frac{s_1 \times \cdots \times s_N}{N}$ Where s_N is security attribute of measure j
Attribute	$s_N \rightarrow s_N^*$ Where S_N^* is the scaled attribute
Security Rating	$\text{Security Rating} = \frac{\sum \text{factors associated with high security}}{\sum \text{factors associated with low security}}$

N is the total number of security attributes for measure j of factor i

Fig. 2 Security Rating Formulation

The overall security rating equation (Equation 7) divides the resilience factor (factor associated with high security) by the factor of threat likelihood and consequence factors (factors associated with low security).

$$SR_a = \frac{F_{Ra}^{\alpha}}{\left(F_{TLa}^{\delta} + F_{Ca}^{\lambda}\right)} \tag{7}$$

Where SR_a is the Security Rating for infrastructure a; F_{TLa} is the threat likelihood factor of infrastructure a; α is the exponential weight of the resilience factor; F_{Ca} is the consequence factor of infrastructure a; δ is the exponential weight of the threat likelihood factor; F_{Ra} is the resilience factor of infrastructure a; and λ is the exponential weight of the consequence factor.

The security rating can be placed on a scale and interpretations made as seen in Figure 3 and Table 1 which are for illustrative purposes for the case study (Dojutrek et al, 2014). The scale and cut-offs can be established at any specific agency to suit their policies.

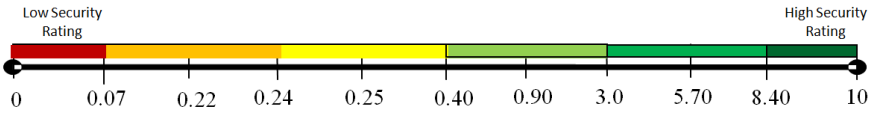


Fig. 3 Security Rating Scale

Table 1 Interpretation of Security Rating

Security Rating	Example Interpretation
≤ 0.21	Indicates a great need for security improvement of the infrastructure. The infrastructure has generally very low security thus immediate action should be undertaken to enhance its resilience and thus to reduce the possible consequences of threats.
0.21 – 0.25	Indicates significant need for security improvement needs of the infrastructure. For this infrastructure, the agency should be poised to undertake actions in the very near future, to enhance resilience and thus to reduce possible consequences of the infrastructure failure.
0.25–0.40	Indicates medium-to-high security improvement needs. Facilities within this range can be monitored at a frequency slightly exceeding standard frequency. The risk of failure can be tolerated until a normal capital project (to enhance resilience and thus reduce consequences, among other benefits) is carried out.
0.40–0.95	Indicates low-to-medium security improvement needs. Unexpected failure can be avoided during the remaining service life of the infrastructure by performing standard scheduled inspections with due attention to specific design features that influence the infrastructure possible consequences.
0.95–3.03	Indicates low security improvement need. Often reflective of the likelihood of threat to a civil engineering system built to the current design standards in a low threat likelihood environment.
3.03–10	Indicates little or zero security improvement needs.

3.2 Fuzzy Logic Framework

A fuzzy logic framework for subjective fuzzification (Figure 4) of measures and attributes for resulting fuzzy output factors is useful when decision makers do not have access to infrastructure specific information for each factor. This method inputs fuzzy data into the security rating equation to find a fuzzy output. Matlab Fuzzy Toolbox was used to program the framework (MathWorks, 2013). For example, each factor can be fuzzified to output a level of that specific factor as seen for the resilience factor in Figure 5. Each measure has a degree of membership ranging from low to high on a determined scale. The value of the resilience factor depends on the level of each measure and the measure levels are determined by respective attributes. Each fuzzified factor value is then input into the overall security rating fuzzy system that results in a fuzzy security rating for a specific infrastructure (Figure 6).

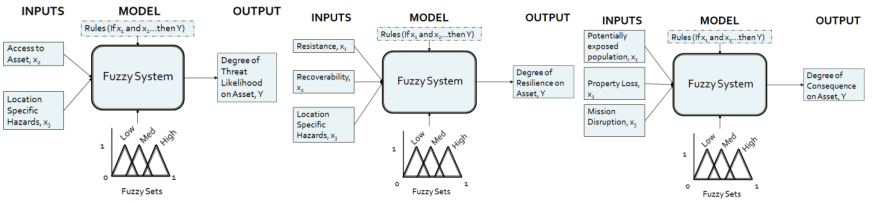


Fig. 4 Fuzzy Logic Models of Security Rating Factors

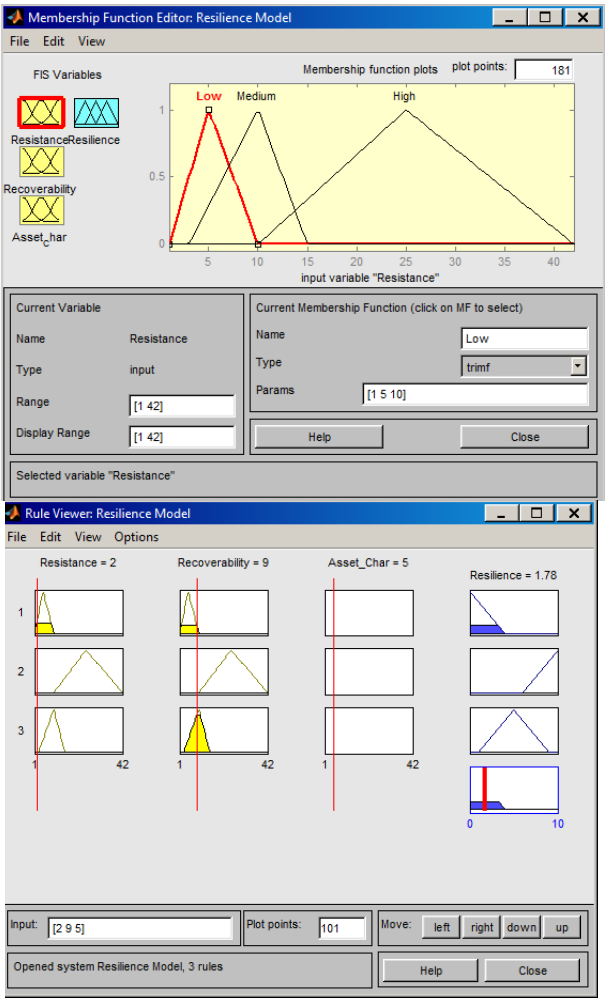


Fig. 5 Fuzzy Consequence Factor and Attributes

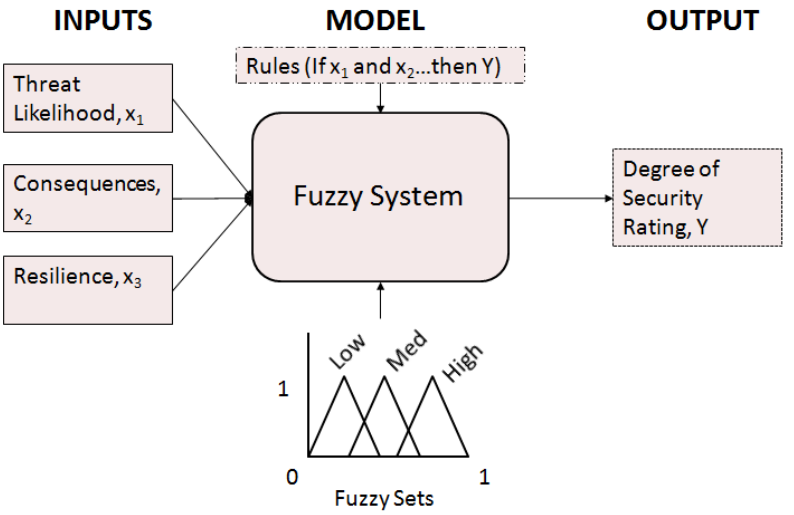


Fig. 6 Fuzzy Security Rating

3.3 Rules

Fuzzy rules were developed to determine the fuzzy security rating output for the fuzzy logic system. The rules give mathematical meaning to the different linguistic levels of each factor in the security rating framework. Thus, a complete fuzzy inference system is created. Fuzzy membership functions for the security rating are shown in Figure 7.

Rules:
If *resilience* is high, *consequence* is low, and *threat likelihood* is low, then *SR* is high
If *resilience* is high, *consequence* is high, and *threat likelihood* is high, then *SR* is medium
If *resilience* is high, *consequence* is medium, and *threat likelihood* is medium, then *SR* is medium
If *resilience* is medium, *consequence* is medium, and *threat likelihood* is medium, then *SR* is medium
If *resilience* is medium, *consequence* is low, and *threat likelihood* is low, then *SR* is medium
If *resilience* is medium, *consequence* is high, and *threat likelihood* is high, then *SR* is low
If *resilience* is low, *consequence* is medium, and *threat likelihood* is medium, then *SR* is low
If *resilience* is low, *consequence* is high, and *threat likelihood* is high, then *SR* is low
If *resilience* is low, *consequence* is low, and *threat likelihood* is low, then *SR* is medium

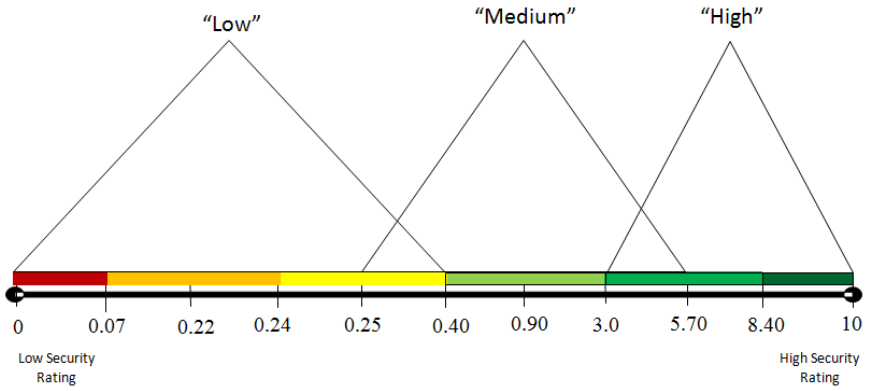


Fig. 7 Fuzzy Membership Functions

4 Case Study

To demonstrate the study methodology, the National Bridge Inventory structure number B05015800100000, the Leo Frigo Memorial Bridge in Brown County, Green Bay, Wisconsin was used (Figure 8). Data was gathered from the National Bridge Inventory dataset available online.

Fig. 8 Leo Frigo Memorial Bridge, Green Bay, Wisconsin

The factors, measures, and attributes used for the case study are described in Figure 9.

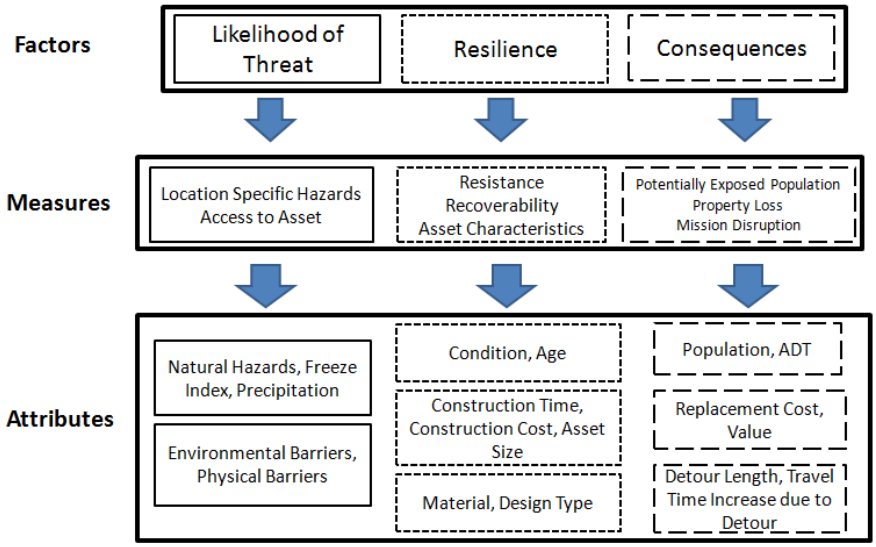


Fig. 9 Detailed Framework for Case Study

A number of assumptions were made for the case study. First, the construction time was based on the bridge size. Second, environmental barriers were assumed to be the waterway under the bridge. The detour travel speed was assumed to be 45mph and all weights in the security rating equation (α, δ, λ) and measures equation were assumed to be equal. Threat likelihood measures, attributes, and scales can be seen in Table 2. The result of each measure is the scaled attributes multiplied together and normalized by the number of attributes for each measure, then multiplied by the measure's weight. Measure weights were assumed to be equal, therefore a value of one was used. After the results are input into the fuzzy threat likelihood factor system, a fuzzy degree of threat likelihood is determined.

Table 2 LFM Bridge Threat Likelihood Factor Data

Measure	Attributes	Data	Scaled	Results
Access to Infrastructure	Env Barriers	Over Fox River	3	3
	Physical Barriers	Independent bridge protection	2	
Location Specific Hazards	Natural Hazards	High Winds, Fog	4	2.66
	County Freeze Index	189.3	2	
	County Precipitation	29.52	1	

The resilience measures, attributes, and scales are listed in Table 3. After the results are input into the fuzzy resilience factor system, a fuzzy degree of resilience is 1.53.

Table 3 LFM Bridge Resilience Factor Data

Measure	Attributes	Data	Scaled	Results
Resistance	Condition	Deck: 8	1	2
		Superstructure: 7	1	
		Substructure: 6	2	
	Age	35 yrs	4	
Recoverability	Const. Time	3yrs	3	9
	Const. Cost	$6.85M	3	
	Infrastructure Size	39,115 ft^2	3	
Infrastructure Characteristics	Material	Steel	2	5
	Design Type	Thru-Arch	5	

The consequence measures, attributes, and scales are listed in Table 4. After the results are input into the fuzzy consequence factor system, a fuzzy degree of consequence is 1.78.

Table 4 LFM Bridge Consequence Factor Data

Measure	Attributes	Data	Scaled	Results
Potentially Exposed Population	Population	Green Bay: 104,868 Brown County: 253,032	4	6
	AADT	31,400	3	
Property Loss	Replacement Cost	$6.92M	3	3
	EDMC Value	$4.34M	2	
Mission Disruption	Detour Length (miles)	~6 miles	2	4
	Inc. in travel time due to detour	8 min	4	

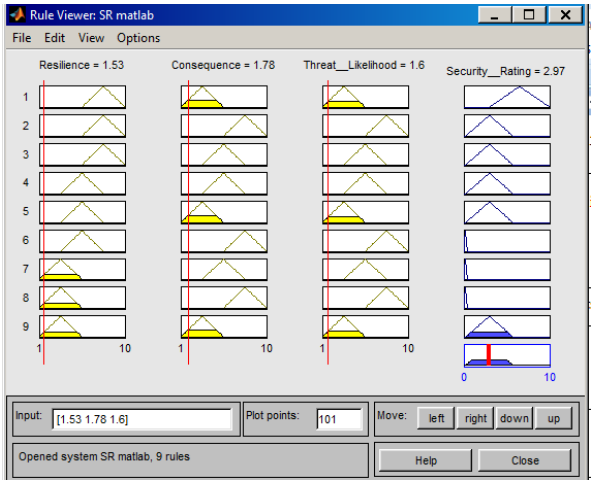


Fig. 10 Overall Fuzzy Security Rating

Each fuzzy factor value was then input into the fuzzy security rating system to result in a fuzzy security rating output for the Leo Frigo Memorial Bridge of 1.6. The overall fuzzy Security Rating of the Leo Frigo Memorial Bridge is then 2.97, which corresponds to a security rating of "medium" as shown in Figure 10.

5 Conclusion and Discussion

Previous literature either did not consider the dynamic or uncertain nature of security data and factors or focused on a terrorist perspective. Threats also include a natural element, such as earthquakes, floods, hurricanes, climate changes, etc. A method that is adaptable to both natural and man-made threat perspectives would require less initial work on the decision-makers' part. Additionally, a method that can transform qualitative information into a quantitative form would be useful to help prioritize among infrastructure for security funding allocation purposes.

This paper first presents a framework to quantify security based on a metric that includes the key factors of risk (threat likelihood, infrastructure resilience, and consequence). These factors have an inherently dynamic and uncertain nature which creates difficulty in accurately predicting their values. Therefore, a methodology to quantify these principal security components through a qualitative method of fuzzy logic was further developed. A qualitative method will enable decision-makers to make decisions about the relative magnitudes of these difficult to quantify security variables. Each security factor was fuzzified using "high," "medium," and "low" levels of its respective measures and membership functions. The fuzzified factors were then input into a fuzzy security rating framework that output the resulting fuzzy security rating for specific infrastructure. A fuzzy system captures the dynamic and uncertain nature of each security factor by creating a fuzzy set of numbers for each level of membership.

The Leo Frigo Memorial Bridge in Green Bay, Wisconsin, U.S.A. was used as a case study for the framework. Data were taken from the United States National Bridge Inventory database to use as an example for determining security measure levels and membership functions for each security factor. All attribute data was scaled and the respective measures fuzzified for input into the overall fuzzy security rating framework. Based on the output, the Leo Frigo Memorial Bridge resulted in a "medium" security rating of 2.97. The case study illustrated how the fuzzy security rating can be determined accounting for the dynamic and uncertain nature of the data.

References

1. Barker, K., Ramirez-Marquez, J.E., Rocco, C.M.: Resilience-based network component importance measures. Reliability Engineering & System Safety 117, 89–97 (2013)
2. Steffey, D.L.: Homeland Security and Transportation Risk. Encyclopedia of Quantitative Risk Analysis and Assessment. John Wiley & Sons, England (2008)
3. Polzin, S.E.: Security Consideration in Transportation Planning. STC White Paper. Center for Urban Transportation Research, University of South Florida (2012) (accessed December 10, 2012)
4. Lundquist, E.H.: International Port Security Program. Defense Media Network (2011), http://www.defensemedianetwork.com/stories/international-port-security-program/ (March 17, 2011)
5. AAR. Class I Railroad Statistics. Association of American Railroads (May 10, 2012)
6. Herbst, B.: Airline Industry-Year 2011 (2012), http://www.airlinefinancials.com/uploads/Airline_Industry-Year_2011ReviewOutlook.pdf, AirlineFinancials.com (March 30, 2012)
7. Labi, S.: Introduction to Civil Engineering Systems. Wiley, Hoboken (2014)
8. Harris, S.P., Dixon, D.S., Dunn, D.L., Romich, A.N.: Simulation Modeling for Maritime Port Security. Journal of Defense Modeling and Simulations: Applications, Methodology, Technology 10(2), 193–201 (2012)
9. Ezell, B.C., Farr, J.V., Wiese, I.: Infrastructure risk analysis model. ASCE Journal of Infrastructure Systems 6(3), 114–117 (2000)
10. INGENIUM, International Infrastructure Management Manual. International Edition, Version 3.0, INGENIUM, Thames, New Zealand (2006), http://www.nams.org.nz/International%20Infrastructure%20Management%20Mannual
11. Hooper, R., Armitage, R., Gallagher, A., Osorio, T.: Whole-life Infrastructure Infrastructure Management: Good Practice Guide for Civil Infrastructure. CIRIA, London (2009)
12. Dojutrek, M.S., Labi, S., Dietz, J.E.: A Multi-criteria Methodology for Measuring the Resilience of Transportation Infrastructure. International Journal of Disaster Resilience in the Built Environment (2014)
13. Dojutrek, M.S.: A Stochastic Multi-criteria Assessment of Transportation Security Investment (working doctoral dissertation), Purdue University, W. Lafayette, IN (June 2014)

14. Melnick, E.L., Everitt, B.S.: Encyclopedia of Quantitative Risk Analysis and Assessment. John Wiley & Sons, West Sussex (2008)
15. Moteff, J.: Risk Management and Critical Infrastructure Protection: Assessing, Integrating, and Managing Threats, Vulnerabilities and Consequences. Congressional Research Service. The Library of Congress (2005)
16. McGill, W.L.: Critical Infrastructure and Portfolio Risk Analysis for Homeland Security. Ph. D. Dissertation. University of Maryland. Department of Civil and Environmental Engineering. College Park, MD (2008)
17. SAIC, Reducing Security Risk for Transportation Management Centers. Presented at the 84th Annual Research Board Meeting (2005)
18. Venna, H.R., Fricker, J.D.: Synthesis of Best Practices in Transportation Security, vol. I. Vulnerability Assessment, Joint Transportation Research Program, Indiana Department of Transportation and Purdue University, West Lafayette, IN (2009)
19. Xia, J., Chen, M., Lie, R.: A Framework for Risk Assessment of Highway Network. Presented at the 84th Annual Transportation Research Board Meeting (2004)
20. McGill, W.L., Ayyub, B.M.: Multicriteria Security System Performance Assessment Using Fuzzy Logic. Journal of Defense Modeling and Simulation Applications, Methodology, Technology 4(4), 1–21 (2008)
21. Yazdani, M., Alidoosti, A., Basiri, M.H.: Risk Analysis for Critical Infrastructure Using Fuzzy TOPSIS. Journal of Management Research 4(1), 1–19 (2012)
22. Yang, Z.L., Bonsall, S., Fang, Q.G., Wang, J.: Maritime Security-Assessment and Management. Journal of International Association of Maritime University 5(1), 56–72 (2007)
23. Yang, Z.L., Wang, J., Bonsall, S., Fang, Q.G.: Use of Fuzzy Evidential Reasoning in Maritime Security Assessment. Risk Analysis 29(1), 95–120 (2009)
24. MathWorks, MATLAB and Fuzzy Toolbox Release, The MathWorks, Inc., Natick, Massachusetts, United States (2013)

A Verification Approach from MDE Applied to Model Based Systems Engineering: xeFFBD Dynamic Semantics

Blazo Nastov, Vincent Chapurlat, Christophe Dony, and François Pfister

Abstract. Model Based System Engineering (MBSE) is "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [1]. Among other principles, it promotes creating and analyzing models all along systems engineering. These models are used to discuss, to argue and finally to make decisions that impact the achieved system (in terms of functioning, costs, safety, etc.). One of the main expectations of MBSE is to permit engineers to dispose of models with a high level of confidence. For this purpose, several model Verification and Validation (V&V) approaches exist, aiming to ensure models' quality in terms of construction (models are correctly built) and in terms of relevance for reaching design objectives and stakeholders' requirements. This paper aims at discussing and evaluating an approach originally developed in the field of Model Driven Engineering by proposing some adaptations. The approach is illustrated on a well-known functional modeling language dedicated to MBSE field.

1 Introduction

Systems Engineering (SE) [2] is a process and a system thinking oriented approach for designing complex systems. In this field, Model Based System

Blazo Nastov · Vincent Chapurlat · François Pfister
LGI2P, Ecole des Mines d'Alès, Parc Scientifique G. Besse, 30000 Nîmes, France
e-mail: {Blazo.Nastov,Vincent.Chapurlat,
Francois.Pfister}@mines-ales.fr

Christophe Dony
LIRMM, 161 rue Ada, 34392 Montpellier, France
e-mail: dony@lirmm.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_16

Engineering (MBSE) is considered as "*the formalized application of* [system] *modeling"* [3] all along engineering activities requested in SE processes [1]. A model represents a system under study so called System of Interest (SOI), considering it under one of the various possible views e.g. functional, behavioral, physical, requirements, contextual, etc. Once created, models allow system engineers to argue and make architectural decisions. These decisions have impact on the realized system, its functioning, its safety, induced costs and so on [4]. Therefore, engineers should have a high level of confidence in models they are handling, before making any decision based on them. So first, models have to be "well-constructed" (correctly built, respecting (meta) modeling requirements and building rules) and second, they have to be the "right models" in terms of 1) relevance for reaching design objectives and 2) respect to a part of, or all, stakeholder's requirements. Moreover, the level of confidence increases if models verify their "well-constructiveness" and "model-rightness", considered separately and in interaction with other models of the SOI.

Classically, models are the subject of study of Model Driven Engineering (MDE) [5] and nowadays they are built using Domain Specific Modeling Languages (DSMLs). So first and foremost is the creation of DSMLs, a process, defining the languages' abstract and concrete syntaxes. An abstract syntax is provided by a metamodel representing, through a graph of classes and associations, the concepts of a domain and their relations. A concrete syntax defines how instances of abstract syntax concepts (forming a model) are concretely represented in a textual or graphical form. In MBSE context, we focus on graphical representation of models. So proposing the abstract syntax and a graphical concrete syntax of a DSML makes it operational to create models, each of them seen as a graphical representation of a particular point of view of a SOI. The literature highlights several frameworks for creating DSMLs and graphical editors [6], [7], [8]. Unfortunately, such editors are of "two dimensional drawing-board" nature because they do not deal with model well-constructiveness or rightness. As a consequence, created models turn-out to be "simple two dimensional drawings" of modeled concepts and relationships.

The main idea of this work is to add a third dimension, allowing designers to create models that can be simulated and questioned, achieving some of the Verification and Validation (V&V) objectives and to ensure the coherence between all models of the same SOI. Some solutions are proposed in the field of MDE [9], [10], [11], but unfortunately they remain, not or hardly applied in the field of SE due to their insufficiency for reaching SE objectives. In this paper, we discuss and evaluate an existing approach coming from MDE by applying it on a DSML, often considered by systems engineers and architects as a relevant functional description language named eFFBD [12] in order to emerge an extension that reaches SE objectives.

The paper is structured as follows. Section 2 discusses the importance of DSML semantics and introduces approaches and concepts that allow defining such semantics. Some limitations of studied approaches are discussed and afterwards an approach exceeding discussed limitations is proposed. Section 3 evaluates that

approach by applying it on the eFFBD language in order to create an executable extension. Section 4 proposes contributions that allow the adaptation of the approach in the field of SE, before concluding about research perspectives in Section 5.

2 Towards Execution and Validation of DSMLs

Abstract syntaxes of DSMLs partially define language semantics through their underlying structure and the vocabulary naming concepts and relationships. Unfortunately, such semantics may sometimes be ambiguous, since different engineers may have different understanding of a single model. Therefore, in order to have equal and non-ambiguous understanding, it is essential to define in a precise and non-ambiguous manner DSMLs semantics.

Semantics are either *static*, independent of any behavior, or *dynamic*, describing the dynamic comportment of models' elements (can be advisedly called "dynamic model" or "dynamic comportment" or "DSML behavior"). There are three ways to formalize dynamic semantics description. First, *operational semantics* describes model comportment as a sequence of states, transitions between states and a machine that executes such a state model. Second, *denotational (translational) semantics* transforms DSML concepts into other DSML concepts with predefined dynamic comportment. Last, *axiomatic semantics* describes in a declarative way the evolution of model properties [13]. In this paper we focus on defining DSMLs behavior using dynamic semantics.

Literature highlights several approaches and tools for defining dynamic semantics for a given DSML. For instance, Kermeta [9] is an executable metamodeling language that defines operational semantics for a given DSML (in imperative way). Another example is the Atlas Transformation Language (ATL) [14] that (in declarative way) defines operational semantics through endogenous transformations and denotational semantics through exogenous transformations. Additionally, metamodeling languages together with constraints definition languages can be used to define axiomatic semantics. Meta Object facilities (MOF) [15] is usually used to define metamodels and OCL (Object Constraint Language) [16] to add constraints to metamodel e.g. pre and post conditions, invariants and so on. However, these tools and approaches are related to software engineering and programming languages which somehow make them difficult to use for SE experts. Indeed, dynamic semantics of dedicated DSML is to be described and formalized with minimal efforts from experts by assisting them and automating the process as much as possible.

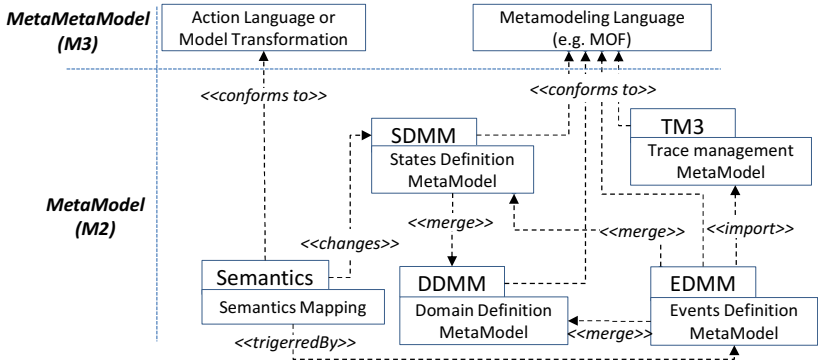


Fig. 1 The executable DSML Pattern [11]

Another approach, supporting state-based execution of DSMLs is proposed in [11]. The approach is schematized in Figure 1 as a pattern composed of four structural parts related to each other and of a fifth part providing the dynamic semantics relying on the previous four. Modeling concepts and relationships between them are defined in the *Domain Definition MetaModel (DDMM)* package. The DDMM does not usually contain execution-related information. Such kind of information is defined in the *State Definition MetaModel (SDMM)* package through several sets of states assigned to DDMM concepts that can evolve during execution. Model execution is described as successive state changes of DDMM concepts provoked by stimuli. The *Event Definition MetaModel (EDMM)* package defines different types of stimuli (events), together with their relationship to DDMM concepts and SDMM states. Applied stimuli are either injected by the environment (exogenous kind) or produced internally by the system in response to other stimuli (endogenous kind). The *Trace Management MetaModel (TM3)* provides monitoring mechanism of model execution by capturing stimuli. The last and key part is the package *Semantics*, composed of evolution rules, describing how the running model SDMM evolves over DDMM concepts (changing their state) according to the stimuli defined in the EDMM. Evolution rules can be either defined as operational semantics or as denotational semantics.

3 Application in the Field of SE

This section proposes to extend the eFFBD functional modeling language by building and tooling its dynamics in order to become able to interpret and animate eFFBD models, and to prove their evolution rules. Similar solution has been proposed by [17] using translational semantics, transforming eFFBD models into Petri Nets models. [18] argues that translational semantics solutions limit V&V objectives. Fortunately, such limitations are exceeded by operational semantics that allow achieving V&V objectives on a source model rather than a third party

model. So the expected result is xeFFBD i.e. an executable eFFBD that we consider as "self-sufficient" in achieving these V&V objectives. In [19] a short history and various evolutions of a particular DSML named FFBD (Functional Flow Block Diagram) and its main evolution named eFFBD (enhanced FFBD) is presented. It is considered as a functional-modeling language preferred in the SE community [20]. This DSML provides system designers with a framework to describe the behavior of complex, distributed, hierarchical, concurrent and communicating systems [21]. For instance, Figure 2 shows the functional architecture of an interface highlighting parallelism, selection, loop and other complex constructs allowed in eFFBD.

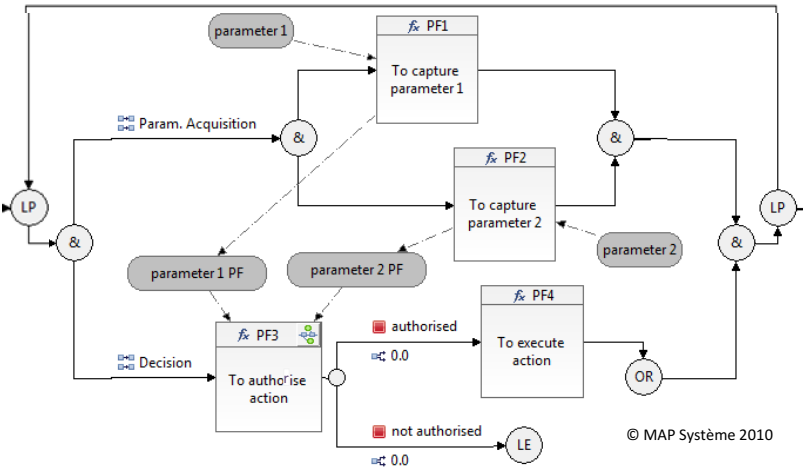


Fig. 2 eFFBD example

Our proposition to achieve executable eFFBD along the above discussed methodology is presented in the following section.

Creating an executable DSML is divided into two major phases. First, a language executable metamodel is defined containing domain (in DDMM), execution-related (in SDMM and EDMM) and monitoring (in TM3) information. Second, semantics are defined describing how execution-related information evolves over domain concepts.

3.1 Phase 1: Executable Metamodel Definition

Executable metamodel definition phase includes four construction stages: DDMM definition, SDMM definition, EDMM definition and TM3 definition. We apply such process to eFFBD as shown in Figure 3.

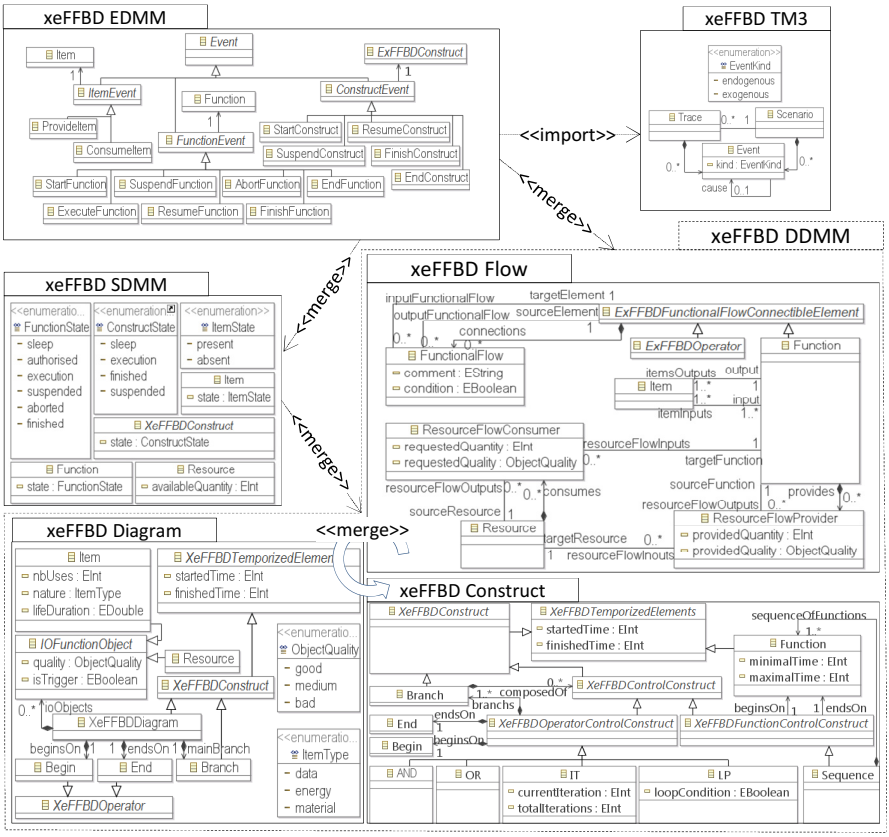


Fig. 3 Creating executable metamodel for eFFBD - construction stages

First, language or domain concepts and relationships are defined. This construction stage is identical to a (non-executable) DSML metamodel definition. The created metamodel is called DDMM since it contains domain concepts and relationships. In order to reduce language complexity and to improve readability and understandability, we split the DDMM into three packages: *xeFFBD Diagram*, *xeFFBD Construct* and *xeFFBD Flow*, each one representing a different aspect of the eFFBD language. *xeFFBD DDMM* is then created by merging all three packages, using the "merge" predefined package operator of MOF.

The xeFFBD language defines three kinds of core elements: Function, Resource and Item. Function describes what a system must do. They transforms one or more input Items in one or more output Items respecting transformation rules, possibly under control of triggers. Resource is something (data, material or energy e.g. human operator, consumable, plans, etc.) that is requested and utilized or consumed during an inputs/outputs transformation. Requested resources are considered as independent from transformation goal and they are requested for function execution that modifies them. Item is something (data, material or

energy) that is requested and transformed by function in order to provide another(s) distinct Item(s). Taking into account its type, an Item can be consumed or can remain available during certain time duration after which its value becomes obsolete and unusable. These core elements are characterized by temporal attributes e.g. minimal and maximal time of execution, life time, etc.

xeFFBD Diagram is the core package describing a xeFFBD diagram as a quadruplet of begin and end operators, main branch and set of input/output objects carried by flows. Begin and end describe starting and finishing points in a diagram. The branch is composed of several control constructions named exFFBD Constructs, described hereafter. Two sorts of input/output objects are then available: items and resources respectively carried out by item flows and resource flows as detailed below. Last, a diagram is temporized element, having started and finished execution time.

xeFFBD Construct package represents different constructions recurring into a xeFFBD Diagram. These constructions allow engineer to describe how functions are chained and the different manners of their execution, introducing the possibility to describe function parallelism, sequence, exclusion, and selection. A construct can either be 1) a function control construct composed of a set of functions (eventually one unique function) put in a sequence, or 2) an operator control construction containing minimum one branch beginning on a begin operator and ending on an end operator. Four types of operator control construction are introduced: AND, OR, Iteration and Loop. A fifth one, named replication construction, is not considered at this moment. AND and OR constructions contain minimum two branches and they represent respectively parallel and exclusive execution of branches. Iteration and Loop constructions represent two possibilities of repetitive execution of one branch differing in the stop condition. Iteration fixes a number of iterations, while loop stops on a Boolean condition. Constructions are temporized elements having started and finished execution time.

xeFFBD Flow package describes three types of flows that can be represented in a xeFFBD model: functional flow, item flow and resource flow. A functional flow describes the order in which functions are executed (related to the primitive relation successor/predecessor between two functions). It is represented by the functional flow class connecting functional flow connectable elements which are either operators or functions. A *Resource Flow* describes requested Resources of a function that consumes them and restores them after execution, modifying eventually some of resource characteristics such as its quality and quantity levels. For this a Resource Flow is characterized by two attributes: *quantity* and *quality*. *Quantity* attribute indicates the requested amount of resource, consumed as an input by a function in order to execute it (*requested quantity*), and provided as an output after execution of related functions (*provided quantity*). *Quality* attribute indicates the level of resource quality, requested as an input in order to execute related functions (*requested quality*), and restituted after function execution as an output altering then eventually the level of quality of the resource (*provided quality*) *i.e.* mixing for instance its availability and its efficiency. Item flow relates

Item with function by input or output relationships. These relationships describe items that are needed and consumed as inputs for function execution and items that are provided as output after execution. Provided items are a result from transformation of inputs flows and eventually under the help or the control of resource flows. Note that there is a special kind of triggering items and resources that can trigger function execution, controlling then function start and/or stop conditions. Functional and resource flow have attributes (comment, condition and quantity, etc.), so they are represented in the metamodel using the class-association pattern, while item flow is represented using associations. Once a DDMM is defined, the second construction stage consists in defining SDMM. In this stage, we define in the package xeFFBD SDMM, the possible states of some of previously defined domain concepts. First, domain concepts that may evolve have to be chosen. For instance, we chose the following: Construct, Function, Item and Resource. The third construction stage consists in defining the events requested for the evolution of evolving concepts together with their relationship with corresponding concepts. Such information is defined in the EDMM. For instance, we defined three types of events: construct event, function event and item event.

Additionally, Figure 4 shows finite states automate associated to the concept Function. Here, on the one hand, function states are represented by automates' states and on the other hand, different types of function event are represented by automates' transitions. The evolution of concepts is represented as transition firing. In this sense, we consider here the input/output transformation described by a Function, is first possible (*Authorized)* i.e. the function can start but wait for Items (and eventually Resources) before being able to make the real transformation of energy, material and / or data (*Execution*) providing then the outputs items and resources (*Finished*). Due to external events, a function can be suspended and even aborted (*Suspended*, *Aborted*) e.g. in case of dysfunction of the component on which the function has been allocated.

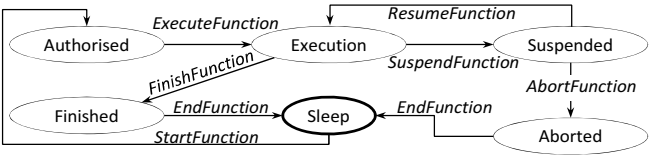


Fig. 4 A finite states automate for the concept Function representing state evolution as transition firing

In the case of Item and Resource, the state model is replaced by defining state variables named quantity and quality allowing us to reduce the number of possible states. For example, a resource of oil can be in state 50 liters, but also in state 100 liters if a function provides as output another 50 liters of that same resource.

The executable metamodel definition process ends by defining a monitoring mechanism considered as fourth stage of this process. For this, we propose the

generic trace mechanism described in the approach, the TM3 package, shown in Figure 3.

3.2 Phase 2: Semantics Definition

As previously defined, xeFFBD metamodel contains execution-related (dynamic) information (e.g. packages SDMM and EDMM). Yet, xeFFBD metamodel is static, until the package Semantics (Figure 1) is defined. This package defines how and when dynamic information is executed on domain concepts, allowing state and property changes. For this, we adopt a proposed property-driven approach detailed in [22]. This approach describes how to formally define execution rules under the form of properties (described below), and how to become able to check some of those properties. Three types of properties can be expressed: structural properties, temporal properties and quantitative properties. The approach distinguishes properties checked on each model execution called universal properties from those checked once called existential properties.

Model evolution is first, defined through universal and existential properties by preconditioning events, second, through transitions that are defined between domain concepts states and finally, through event-based transition firing. When fired, transitions invoke state changing of domain concepts. Figure 4 illustrates different states of Function, event-based transitions between states and corresponding events. For instance, if event *StartFunction* is applied on an instance of *Function* that is in state *Sleep*, a transition is fired changing its state into *Authorized*.

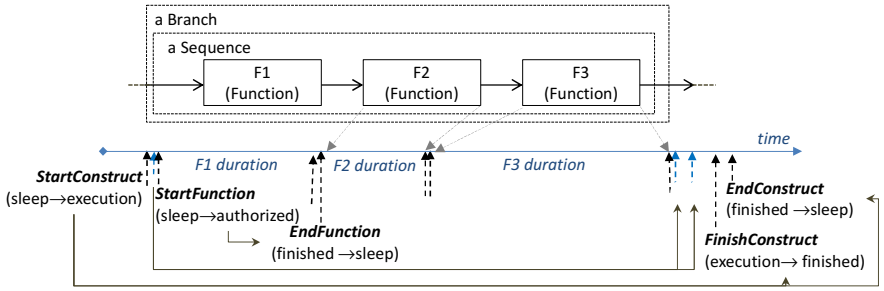


Fig. 5 Execution of a simple exFFBD model

We consider that execution of lower level embedded constructions is controlled (*i.e.* started and finished) by higher level embedding constructions respecting an ordering given by a functional flow. Figure 5 illustrates an example of such execution control describing applied events and invoked state changes of components represented by different type of arrows. A simple xeFFBD diagram is represented by a starting point (entering arrow), an ending point (exiting arrow) and a main branch. A sequence is placed inside that branch, containing three functions: F1, F2 and F3. In this example, input and output object flows are not

represented in order to ease the readability of the figure. The execution occurs as detailed hereafter. Each Construct controls the execution of Branches and Constructs it contains. So, the diagram starts the main branch which starts the sequence. Since a sequence contains functions, it also controls their execution. First, it starts the beginning function (F1), and afterwards it waits for finished functions, to end their execution and to start the execution of following functions (end F1 and start F2). This process is repeated until the ending function (F3) ends and then the sequence itself finishes execution. The main branch ends the sequence, before finishing its proper execution. The diagram waits for the main branch to finish execution, in order to end it. The end of the main branch execution means that the diagram can first, finish and then end its execution.

Functions are contained in a sequence, so the formal definition of their execution starting and ending is defined by the dynamic behavior of the Sequence construct as described in previous example. Next, due to lack of space we formally define only a dynamic behavior of functions using the previously described property-driven approach. An input/output transformation described by the Function is first possible *i.e.* the function can start but has to wait for Items and eventually Resources (Figure 6, Eq.1) before being able to make the real transformation of energy, material and / or data (Figure 6, Eq.2) providing then the outputs items and resources and finishing its execution respecting minimal and maximal execution time (Figure 6, Eq.3).

For f ∈ Function

(Eq. 1)	{ (f.state==***authorised***) **AND** (∀ i ∈ f.itemInputs,(i.state==***present***)) **AND** (∀ j ∈ f. resourceFlowInputs, ((j.requestedQuantity >= j.sourceResource.availableQuantity) **AND** (j.requestedQuality == j.sourceResource.quality)))) **implies** ***executeFunction***(f) }
(Eq. 2)	{ (f.state==***execution***) **implies** ((∀ i ∈ f. itemInputs, (***consumeItem***(i))) **AND** (∀ j ∈ f. resourceFlowInputs, (j.sourceResource.availableQuantity **-=** j.requestedQuantity)) }
(Eq. 3)	{ ((f.state==***execution***) **AND** ((internalTime - f.startedTime) >= minimalTime) **AND** ((internalTime - f.startedTime) <= maximalTime)) **implies** (***finishFunction***(f)) }
(Eq. 4)	{ (f.state==***finished***) **implies** ((∀ i ∈ f. itemOutputs, (***provideItem***(i))) **AND** (∀ j ∈ f. resourceFlowOutputs, (j.targetResource.availableQuantity **+=** j.providedQuantity)))}

Fig. 6 Semantics mapping by defining evolution properties of Function concept

Let us note that due to external events, a function can be suspended temporarily, can resume its execution or can abort (*Suspended*, *Aborted*). These external events can be then shared with other constructs from other modeling languages. For instance, the function behavior can depend on the component behavior that performs this function. So, the event Suspended can be a common event shared between xeFFBD and a future DSML named xPBD (executable Physical Block Diagram currently under study).

4 Application Discussion and Expected Contributions

State Notion and Formalization. Considered approach describes concepts' execution as successive state change. This induces to define a set of states considered by the user as sufficient for his V&V objectives. Unfortunately, some concepts, such as Resource from xeFFBD, may be characterized by a continuum of states. For this, we propose three solutions. The first solution consists in defining a finite number of states. For instance, resource states model can be reduced to a two state model, containing: *sufficient* or *insufficient* states. However, this solution is too limitative for V&V objectives. The second solution consists in introducing a set of state variables describing possibly infinite number of states that can evolve continuously. This solution allows describing high level of detail. For instance resource state model can be represented by *quality* and *quantity* variables. The third one consists in mixing the previous two solutions, linking discrete states defined in the state model and state variables. For instance, Item state model is composed of a state quality variable and two states: present and absent. These three solutions are applied in SDMM package and are now under development.

Towards Condition and Event Based Transition Approach. In order to understand concepts' evolution, one has to simultaneously visit three packages: DDMM, SDMM and EDMM, and the evolution rules defined in the semantics package. We consider that this makes created languages difficult to read and understand. In order to ease readability and improve understandability, we propose a representation of previously stated packages, using finite state automata. Conditions and events are responsible for a transition firing. First, a Condition (True by default) is a Boolean function computed on variables, attributes of any concept from the local DDMM and external variables corresponding to other concepts from another DDMM. So *inter* and *intra* conditions are distinguished. Intra conditions have to be satisfied by a currently manipulated model, while inter conditions correspond to conditions that have to be satisfied by one or several other models from the same SOI whose behavior interacts with the behavior of studied model. Second, it is always possible to distinguish two events and there exist a default event *e* always occurring. A Transition can be then fired when associated event is received, and if and only if, associated condition is verified. The work now consists of formalizing these notions and linking the transition behavior with discrete event system theory.

Towards Model Transient States Detection and Management. Temporal evolution rules (named properties by [13]) are currently defined using Temporal OCL (TOCL). This induces the examination of defined properties taking into account a unique scale of time. However, the notion of "*model stability*" is by hypothesis essential for models representing critical, parallel or distributed systems. A "*transient state*" of a concept is a state such that it is possible to change that state without modifying the inputs, as defined for instance in the case

of Sequential Function Chart in [23]. A model is stable if and only if each instance of a modeling concept used in this model is itself in stable state. We propose here to extend considered approach by a double scale of time named external and internal time modeled by two independent logical clocks. Values of each variable v_i appearing in conditions and occurring events associated to transitions of a state model M are read and then frozen in external time. M evolves taking into account v_i by using then an internal scale when performing execution rules allowing then to detect transient states and to reach the next stable state of M. The external time depends on environment evolution and is a logical modeling of physical scale time. It is defined as a set of moments ordered by taking into account events apparition. It is initialized when a simulation starts. The internal time is initialized at each moment defined in external time and there are no common temporal dimensions between internal and external scales. The evolution algorithm allowing transient states detection is schematized in Figure 7.

Towards Properties Modeling Languages and Checking Techniques. Literature highlights several property-driven approaches with associated V&V techniques [24], [25] that will be explored and applied in the prosed frame of work.

Towards Modeling Languages and Models Interoperability. As illustrated in Section 3, we propose to become able to link formally the resulting interpretation and execution of several DMSLs each dedicated to the description and the analysis of a view of a given SOI (behavioral, physical, functional, etc.). This will allow contributing to become able to check the coherence of SOI models even considering different points of view and different modeling objectives. It is a question of linking the dynamic semantics i.e. SDMM and EDMM have to be extended introducing requested and shared concepts and evolution rules.

Tooling. Unfortunately, tools supporting the considered approach do not exist at this moment. An extension of Diagraph [6] is now under development taking into account proposed improvements.

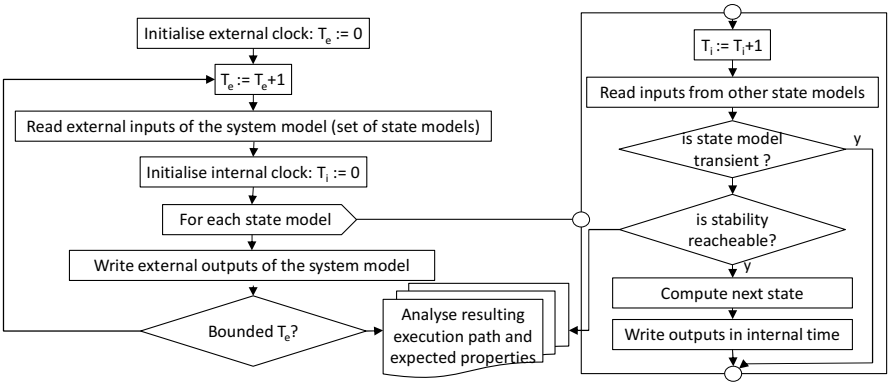


Fig. 7 proposed evolution algorithm including stability reaching objectives

5 Conclusion and Perspectives

This paper presents an approach from the field of MDE for defining semantic of a DSML. The approach is considered here as a formal and relevant way for achieving models V&V objectives. It is applied to a functional modeling language largely used in MBSE domain. This application however, makes appear some questions that seem crucial and remain partially or completely uncovered. Conceptual as technical improvements are then proposed in order to complement this approach. The research and development work is now on going intending to fully support executable DSMLs creation process and to deploy it on MBSE domain.

References

1. INCOSE, Systems Engineering Vision 2020, INCOSE-TP-2004 (September 2007)
2. ISO/IEC, ISO/IEC 1528: Systems and software engineering - System life cycle processes, vol. 2008(1), p. 5. IEEE (2008)
3. Estefan, J.A.: Survey of Model-Based Systems Engineering (MBSE) Methodologies 2. Differentiating Methodologies from Processes, Methods, and Lifecycle Models. Jet Propuls 25, 1–70 (2008)
4. BKCASE Project, System Engineering Book of Knowledge, SEBoK v1.2., `http://www.sebokwiki.org/`
5. Kent, S.: Model Driven Engineering. In: Integr. Form. Methods, pp. 286–298 (2002)
6. Pfister, F., Chapurlat, V., Marianne, H., Nebut, C.: A light-weight annotation-based solution to design Domain Specific Graphical Modeling Languages. In: Proceedings of Modelling Foundations and Applications - 9th European Conference (2013)
7. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, p. 744. Addison-Wesley Professional (2008)
8. Pontisso, N., Chemouil, D.: TOPCASED Combining Formal Methods with Model-Driven Engineering. In: 21st IEEE/ACM Int. Conf. Autom. Softw. Eng. (2006)
9. Jézéquel, J.-M., Barais, O., Fleurey, F.: Model driven language engineering with Kermeta. In: Fernandes, J.M., Lämmel, R., Visser, J., Saraiva, J. (eds.) GTTSE 2009. LNCS, vol. 6491, pp. 201–221. Springer, Heidelberg (2011)
10. Mellor, S.J., Balcer, M.J.: Executable UML: A Foundation for Model-Driven Architecture, p. 416. Addison-Wesley Professional (2002)
11. Combemale, B., Crégut, X., Pantel, M.: A Design Pattern for Executable DSML. In: The 19th Asia-Pacific Software Engineering Conference (APSEC), pp. 282–287 (2012)
12. DoD, Systems Engineering Fundamentals. Def. Acquis. Univ. Press (2001)
13. Combemale, B.: Approche de métamodélisation pour la simulation et la vérification de modèle – Application à l'ingénierie des procédés. Phd - INPT (2008) (in French)
14. Jouault, F., Allilaire, F., Bézivin, J.: ATL: a QVT-like transformation language. In: Companion to 21st ACM SIGPLA, pp. 719–720 (2006)
15. OMG, MOF Core specification, v2.4.1 (2013), `http://www.omg.org/spec/MOF/2.4.1/PDF/`

16. OMG, OCL: Object Constraint Language, v2.4 (2014), http://www.omg.org/spec/OCL/2.4
17. Seidner, C.: EFFBDs Verification: Model checking in Systems Engineering. Pdh University of Nantes (2009) (in French)
18. Chapurlat, V., Braesch, C.: Verification, validation, qualification and certification of enterprise models: Statements and opportunities. Comput. Ind., 711–721 (2008)
19. Haskins, C., Forsberg, K., Krueger, M.: Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities. In: Systems Engineering. INCOSE (International Council on Systems Engineering) (August 2011)
20. Chesnut, H.: Systems Engineering Methods. Wiley & Sons (1967)
21. Aizier, B., Chapurlat, V., Lisy-Destrez, S., Prun, D., Seidner, C., Wippler, J.-L.: xFFBD: towards a formal yet functional modeling language for system designers. In: 22nd Annual INCOSE International Symposium (2012)
22. Combemale, B., Cregut, X., Garoche, P.-L., Thirioux, X., Vernadat, F.: A Property-Driven Approach to Formal Verification of Process Models. In: Filipe, J., Cordeiro, J., Cardoso, J. (eds.) ICEIS 2007. LNBIP, vol. 12, pp. 286–300. Springer, Heidelberg (2008)
23. IEC 60848, Specification language GRAFCET for sequential function charts. 2nd edn. (2000)
24. Dasgupta, P.: A roadmap for formal property verification. Springer (2010)
25. Chapurlat, V.: UPSL-SE: A model verification framework for Systems Engineering. Comput. Ind. 64(5), 581–597 (2013)

How to Boost Product Line Engineering with MBSE – A Case Study of a Rolling Stock Product Line

Hugo G. Chalé Góngora, Marco Ferrogalini, and Christophe Moreau

Abstract. This paper presents the first results of current product line engineering efforts at Alstom Transport. It describes the background that has led us to these efforts as well as the approach that we have adopted on Model-Based Systems Engineering (MBSE) and Product Line Engineering (PLE). We present a real-life application of MBSE and PLE that "goes beyond" SysML along with a quick overview of the first results stemming from this application to Rolling Stock systems.

1 Introduction

The Purpose of Product Line Engineering. As in many other industries, the need for better, safer and more reliable system developed at a lower cost and with ever shorter times-to-market has kept growing constantly in the transportation sector. To cope with this demand, systems and domain engineers strive to master the complexity of their systems while integrating innovative solutions into their products and keeping up with the state-of-the-art in their domains. Besides, in most industries, very few or practically no systems are created "from scratch", so engineers are likely to reuse knowledge from a previous project or product in the form of documents, processes, or models.

For the common folk, the purpose of reuse might seem obvious: to leverage previously developed assets into a new project in order to *improve* project characteristics such as quality, cost effectiveness, time to delivery or risk

Hugo G. Chalé Góngora · Marco Ferrogalini · Christophe Moreau
ALSTOM Transport - RSC Global Engineering, 48 rue Albert Dhalenne,
93482 Saint-Ouen Cedex, France
e-mail: {hugo-guillermo.chale-gongora, marco.ferrogalini,
christophe.moreau}@transport.alstom.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_17

mitigation. This implies that the benefits of reuse or the reuse efficiency in the context of a given project could or should be measured. However, while reuse is a relatively well documented practice in software and manufacturing, the formalization of this practice in the systems engineering domain is relatively new and little has been reported on actual industrial applications, despite a historical, *de facto* or carefree reuse practice in industry (like the utilization of existing specification documents, technical drawings or test procedures) [Fortune and Valerdi].

The concept of reuse is related to strategies such as *platform engineering, product family engineering* or P*roduct Line Engineering (PLE)*, which can be applied to all types of engineering assets. The origins of PLE can be roughly traced back to the mid 1970's with the appearance of the first works on the formalization of software product families. Since then, literature is filled with numerous attempts to define languages and approaches for reuse, variability modelling and PLE. Among the most recent efforts, we can cite the ISO/IEC 26550 standard published in 2013 [ISO], the work carried out by the Duisburg-Essen University [Pohl et al.] or OMG's reusable asset specification [OMG 2005] and its initiative on CVL [OMG 2012] (the reader can refer to [Czarnecki et al.] and references therein for a wider overview of variability modelling approaches). To our knowledge, very few of these works, however, have actually been the object of wide industrial applications. An interesting preliminary report on the application of PLE in the automotive domain is documented in [Flores et al.].

The term Product Line Engineering (PLE) defines a process that helps managing the underlying architectures of the product platforms (or the product portfolio) of an organization in order to maximize the benefits of reuse. In software and product development PLE is extended to *Systems and software product line engineering* and refers to the production of a family of products that share a set of assets and common production facilities. We have adopted the former definition by expanding the term *architecture* to encompass all kinds of structured, organized data used to characterize our systems in their entirety, from operational concepts, through requirements, architectures, validation plans, down to components and parts descriptions or specifications.

In our case, the purpose of adopting Product Line Engineering for rolling stock is clearly to **improve our business (our profitability) by maximizing the benefits of reuse**. We must bear in mind, however, that reusing an asset should be the result of a well-documented decision process and that implementing PLE requires upfront investment and thought. It is safe to postulate that, in most organizations, one could find examples where reusing an asset actually proved to be less profitable over the system lifecycle than developing a new one: while the purchase cost of the asset might have been low, the overall costs induced by debugging, repair, validation, warranty expenses or penalties slowly but surely ended by eating up the originally planned profits. To put it simply, copying solutions from one project to another without an overarching strategy cannot be considered as PLE (or as engineering, for that matter) and it cannot yield by itself all the potential benefits of product line engineering.

Model-Based Systems Engineering. When trying to improve the quality of products or the efficiency of the development cycle, industries usually tend to put in place process improvement initiatives (like processes standardization or the adoption of maturity models) or to adopt model-based engineering, Systems Engineering (SE) and, in some cases, standard architecture frameworks as a support to these initiatives.

Model-Based System Engineering (MBSE) is the application of modelling techniques to support the systems engineering process activities. MBSE can make use of any combination of static or structural models, dynamic or behavior models and executable 1D, 2D or 3D models that take the form of simulations. These models can be used to refine the operational concepts and support the optimization of system architectures [Doufene. et al. 2013, 2014], support trade-off studies, support verification and validation, or promote the development of cohesive operational, functional and physical architectures of the system. On the latter aspect, architectural design frameworks that provide guidance and rules for structuring, classifying and organizing the system architectures are often used when deploying MBSE inside an organization. Such frameworks serve as a reference to organize all the architectural elements describing a system according to several complementary viewpoints. The architectural views defined by the viewpoints are essential to cover the whole scope of the system architecture and to tackle system complexity by representing the system at different abstraction layers.

Modeling and simulation have been in use for many years in different fields to support the development of systems or products with great success. In the systems engineering domain, some arguments arise as to whether there exists something such as "non-model-based" SE. In any case, as cited by [Van Gaasbeek], one can date some early applications of computer-aided tools for model-based systems and software engineering in the defense sector back to the mid 1960's, the theoretical foundations of MBSE to the early 1990's in the work of [Wymore] and, more recently and the effort to model a unified architectural description framework by [Grady] to the late 2000's.

A model-based approach shifts the nature of representation of systems from prose forms to explicit (and theoretically unambiguous, if not formal) data structures and representations, with expected benefits such as improvement in quality and communication, costless traceability, increase of productivity [Estefan 2007, Friedenthal et al. 2008]. MBSE, however, is no panacea and its application and deployment in an organization requires deep reflection and (again) investment. To illustrate this, consider that in order to apply MBSE in an effective manner, all the assets that are used or produced in each of the systems engineering process activities must be formalized in an appropriate way. This can be a huge task when working in an organization with a very large asset legacy captured in *ad-hoc*, inconsistent formalisms and supports.

Finally, one can naturally intuit that combining MBSE and PLE into something like "MBPLSE" (Model-Based Product Line Systems Engineering, as suggested in [AFIS]) could yield even more benefits than each of these practices alone could.

It must be pointed out that part of product line engineering deals with modeling the variability of the engineering assets that constitute the product lines, which introduces yet another abstract viewpoint and formalism that must be integrated into a MBSE framework. So the problem that arises from can be stated as follows: how to combine successfully a MBSE approach with a PLE approach, in order to implement, in real life everyday practice, an efficient and profitable reuse strategy for an organization. This paper presents a practical approach and our first results as an answer to this question.

2 PLE for the Railway Rolling Stock Sector: Origins, Motivations and Generations

Origins Part One – The Railway Rolling Stock. Alstom Rolling Stocks product line bases its offer on a large product family portfolio. The highest-level criteria to distinguish one family from another are basically the maximum speed, the journey distance, the number of passengers and the geographical region (Figure 1).

Fig. 1 Rolling Stock product line chart

Historically, the product definition at Alstom and, more generally, in the railway rolling stock sector has been guided completely by the market demand, except for some isolated single projects for which a product is defined in an *ad-hoc* manner in a "product push" strategy. The adoption of the product line engineering concept is relatively recent and typically comes from product manufacturing sectors like automotive, where the product-push is the more commonly adapted model for their type of market.

In fact, in the range of rolling stock products portfolio, due to the differences between customers and their needs, we can identify one type of products that normally require light customizations (these would be more "aesthetic or style-oriented", like trams or metro products, for instance) and another type of products where the required customization may be very heavy (sometimes, customers

require even a specific train architecture, like high and very high speed trains, for instance). We can then assume that there exists a relationship between rolling stock product speed - journey distance and its capacity to be pushed into the market.

Like stated above, the purpose for adopting the product line engineering approach in the railway rolling stock sector is to maximize the reuse of an existing product in order to rely on service-proven solutions to minimize technical risks, to reduce the project costs and to reduce the time to market.

Origins Part Two – The Challenge of Cultural Background. At ALSOTM Transport and probably like in all companies with a strong mechanical and manufacturing background (like the Railway Transportation industry), we have been (and still are!) confronted to two main challenges associated to PLE. The first challenge is that in the mind of most actors of the project reuse concerns only tangible final products (parts, hardware/software components or systems), just as described by [Wymore and Bahill]. In our case, one of the problems with this point of view is that it overlooks the fact that a reused product is almost inevitably modified during a project, introducing variants that are rarely formalized and retro-fed to the product line baseline. Likewise, engineers "by pass" very often the necessity to perform a thorough, systematic analysis of the new context in which the product or products will be reused. The Arianne 5 catastrophe is probably the most extreme illustration of the consequences of overlooking this latter aspect.

The second challenge concerning PLE is that the activities of the systems engineering processes produce also less tangible products (like architectures, operational scenarios, use-cases, validation plans, justifications of design choices or even tacit knowledge) and reusing them requires upfront reflection and investment to formalize and adapt these products to specific project contexts [Fortune and Valerdi]. We are often confronted to a futile debate as to which has the greater value or which bears the greater potential profits for an organization: reusing physical parts or reusing non-physical engineering assets? We have chosen not to solve explicitly this byzantine dilemma, but rather to let the reader go through the present paper and, hopefully, help him elaborate his or her own opinion on this topic.

PLE is a possible solution for these challenges but, just like MBSE, it is no silver bullet. If it is understood and implemented poorly, the significant investments that are required may result in underachievement of expected benefits.

3 Defining a Reuse Strategy

One of the advantages that we have found when starting to model variability is that we provided the essential elements that will allow evolving from an Opportunistic to a more Strategic approach to reuse. Opportunistic or "bottom-up" reuse is often the default strategy of organizations trying to obtain savings through

reuse but without an overarching strategy to accomplish this. The most advanced examples of opportunistic reuse target component reuse through architectural strategies (e.g. modular components, platforms) and product manufacturing (e.g. flexible production lines). Most engineers (system integrators, in particular) who follow this approach can cite an experience of problematic reuse, mainly because the conditions that must be put in place to achieve success were not considered or even known [Fortune and Valerdi].

Strategic reuse is a more organizational, process-oriented, disciplined approach with greater expected benefits that requires an organization to have a different point of view on reuse. It proposes an effective framework for reuse that can bridge the gap between customer needs or marketing offer and the final train production phase in plants. It focuses on the reuse of all systems life-cycle assets, not only on existing components. Among other things, a strategic reuse approach requires that a mature asset repository exists and that resources are available to tailor assets to a particular project application. Assets to be reused can be pulled from the repository then tailored or prepared (i.e. extracted, analyzed, modified or simply read) for their reuse, but the repository can also be populated with assets developed throughout the systems engineering activities of a given project. That is, as a project evolves, an organization can work in parallel to capture reuse opportunities along the way, such as new components, procedures, methods, plans or tools (this is actually how our first asset repositories are built).

The existence of an asset repository, however, is not enough to manage efficiently a product line. PLE has to take explicitly into account multiple products and the variations within and between them. An upfront, perfectly stable planning of variability is practically impossible because sources of variability can still emerge from, for instance, the arrival to the market of a new technology or from the result of benchmarking (e.g. an unexpected offering from competitors). Nevertheless, the identification of most variability needs should be based on the careful analysis of target markets, the offer portfolios of competitors, the state-of-the-art and other factors. Furthermore, in order to assist the decision making process about reusing or not a given product or asset, their reusability should be characterized in terms of the value that their reuse brings to the project (typically, the degree of reuse versus the expected savings or versus an investment/depreciation ratio). In the software development sector, for instance, reusability has been studied for a long time an yielded several proposals for metrics and capability models, like the Reuse Capability Model [Davis], just to give one example.

Distinction between common and variable parts of members of the product line affects the way in which PLE will be formalized and managed and it also affects the organizational aspects inside a company, in many ways [Flores et al.]. Managing an asset repository, maximizing commonalities amongst assets and mastering the diversity of a product portfolio are the objectives of PLE and variability modelling.

4 Looking Back, Looking Forward

By looking to the past, present and future concepts of how to implement a reuse strategy at Alstom rolling stocks, we can identify three different approaches or "generations" of reuse. These differ basically by the level of maturity and deepness on the understanding and deployment of the product line engineering paradigm. The following paragraphs explain these different generations of reuse strategy maturity.

First Generation Reuse Strategy or "Basic Reuse". In this type of approach, a real product family does not exist. By a real product family we mean a product which has been conceived in order to answer to a wide range of different needs with alternative solutions (variants and options).

The key activity in this type of approach is to identify among all the already manufactured or delivered products which is the closest one to the requirements and needs expressed formally (through a request for proposal or RFP) by a new potential customer (Figure 2).

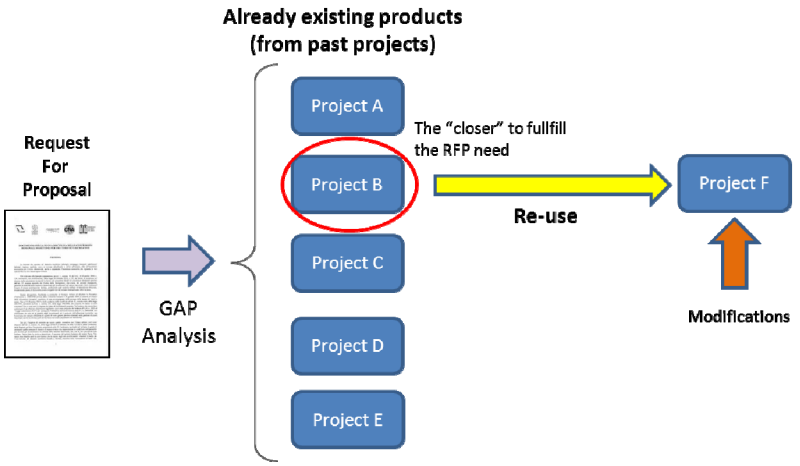


Fig. 2 First generation reuse strategy

Then the selected engineering artefacts of the previously existing product are re-used and modified in order to completely fulfill the RFP requirements through multiple iterations. In this approach the product variability is not designed (or modelled) nor managed.

Second Generation Reuse or "150%" Rolling Stock Product Family. In this type of approach the product is designed as family of products by starting with the definition of the stakeholders visible product characteristics (external variants) and internal product characteristics (internal variants) before going into the definition of all the possible detailed product performances and different alternative solutions. We talk about a 150% product family in the sense that the product

family scope is normally wider than what it would be needed for a single project, considered to be a 100% scope. The product family consists of two levels (system and subsystems), associated through the traceability links declared in the engineering artefacts global meta-model (further details are given in the next section) and trough the product characteristic that are subject to variation (variants) at all levels. The reuse approach is thus recursive across and within the different system levels.

To answer to a new request for proposal, a gap analysis has to be performed in order to identify the set of product characteristics (variants) of the product family that fulfill the needs stated in the RFP and the set of needs that has to be addressed with specific (i.e. out of the product family) custom solutions. Once a coherent and homogenous set of family product characteristics that covers completely or, more often, partially the needs in the request for proposal has been identified, then a specific product will be instantiated from the family and the specific custom solutions (if necessary) will be added (Figure 3).

The product family has then a life-cycle which is independent from all the instantiated projects, but during their lives there might be several synchronization points where all the modifications made at product family level or at the project level can be shared and eventually implemented. The product family can therefore be enriched by the return of experience from the several instantiated projects.

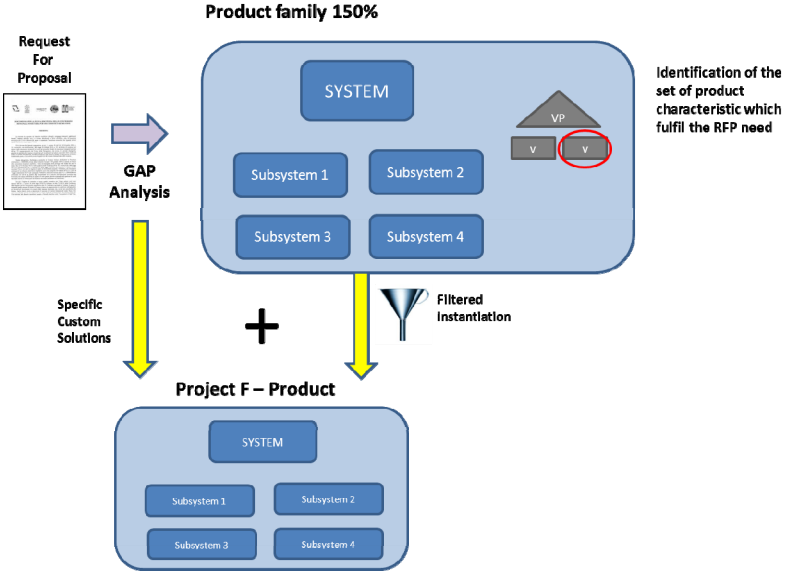


Fig. 3 Second generation reuse strategy

Third Generation Reuse or "System and Subsystems Catalogue of Buildings Blocks". In this type of approach, besides the 150% product family described above, there exists a repository of reusable "modular assets" that stores both the product family assets and the project (or projects) assets.

Just like in the second generation reuse strategy presented above, in order to answer to a new request for proposal a gap analysis has to be performed to identify which parts of the product families at system and subsystem level fulfill the RFP needs and which part of these needs has to be addressed with some specific (out of the platform) custom solutions. The product that covers the needs of the request for proposal will be instantiated from the 150% product families (at system and subsystem level) by choosing the variants to be included in the product, on one hand, On the other hand, the asset repository will be queried in order to identify the relevant product family assets and the returns of investments in them that match the chosen variants and the objectives and constraints of the project, respectively. Once this choice is made, a specific product will be instantiated ("filtered") from the product family and built ("populated") with the pertinent assets, in order to produce a consistent product solution for the project (Figure 4). If necessary, the necessity to develop specific custom solutions to answer to uncovered RFP needs will be identified and the associated project assets will be developed and stored in the asset repository.

These product families have then a life-cycle that is independent from all the instantiated projects, but during their lives, there might also be several synchronization points where all the modifications made at product family level or in the projects can be shared and eventually implemented. The catalog of "modular assets" can therefore be enriched by the return of experience coming from the several instantiated projects.

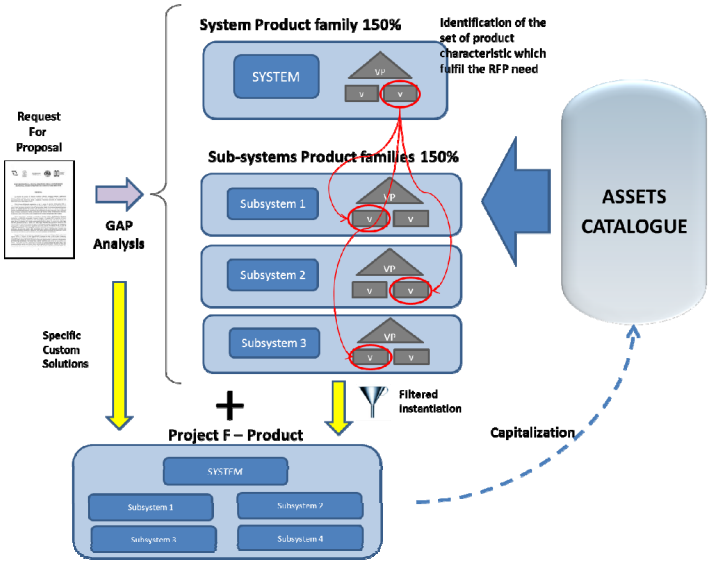


Fig. 4 Third generation reuse strategy

5 Model Based Framework for MBSE and PLE

A Rolling Stock product is a complex system where the complexity is exponentially increasing (more complex functionalities, more integration with other systems). A classical top-down approach (from stakeholder's requirements to product solutions) is therefore strongly required. Furthermore, for the reasons exposed in the previous chapters, the re-use of existing products (bottom-up approach) is mandatory. We can then conclude that the classical system engineering top-down approach has to be fused with the typical re-use bottom-up approach.

To ensure a good quality of the product development process and to enable a painless product re-use, Alstom decided to rely on the Model Based System Engineering approach, which is *"… the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual phase and continuing throughout development and later life cycle phases"* [INCOSE].

The product elements that will be modelled using objects of different semi-formal or formal object-oriented languages are shown in Figure 5 below..

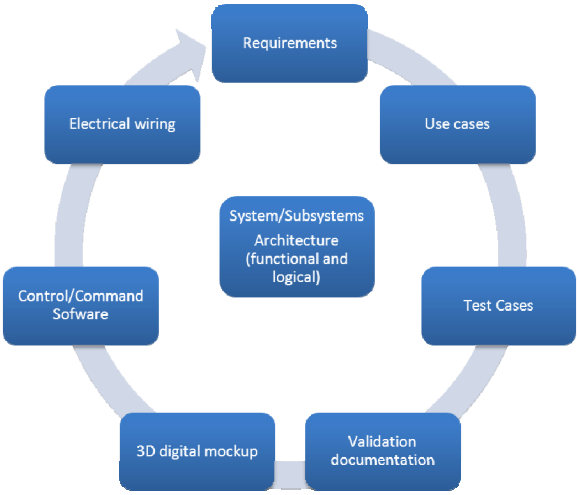


Fig. 5 SE elements modelled in ALSTOM MBSE approach

One of the key advantages of using models is to enable the consistency of the different engineering artefacts. To enable this, Alstom has defined the global traceability meta-model shown in Figure 6.

Since the engineering artefacts are stored in different databases associated to the tools that have been used to model these artefacts, custom interfaces between tools have been (or will be) developed in order to implement a full traceability between artefacts, as shown in the meta-model. Figure 7 below shows a general scheme of the different databases and the links between them.

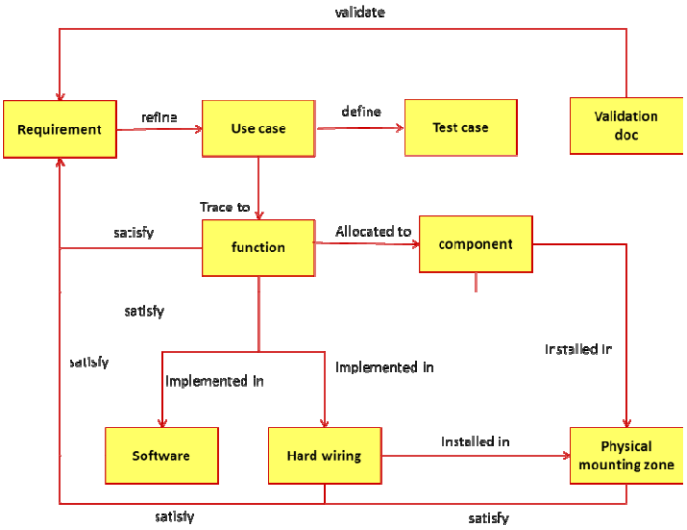


Fig. 6 SE meta-model

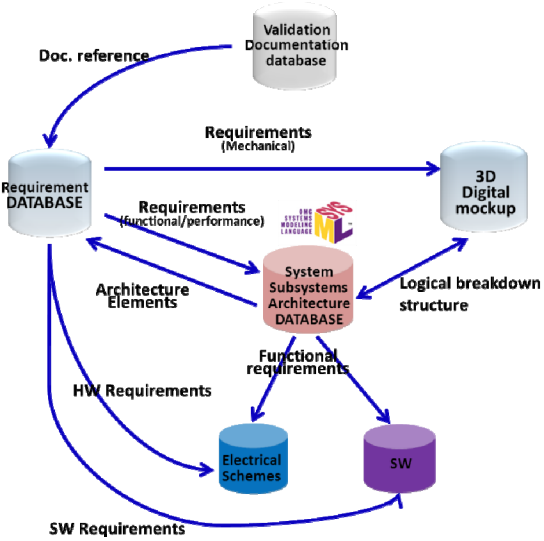


Fig. 7 Links between engineering artefact databases

Second and third generation reuse strategies, as defined in the previous chapter, can be considered as two legitimate product line engineering practices. To enable them, it is necessary to formalize the variability of the product line. The language that we have chosen to model variability and declare the artefact dependencies is named Orthogonal Variability Modelling [Pohl et al.]. Figure 8 shows the basic notation of the OVM language illustrated by a simple example of an energy capitation system.

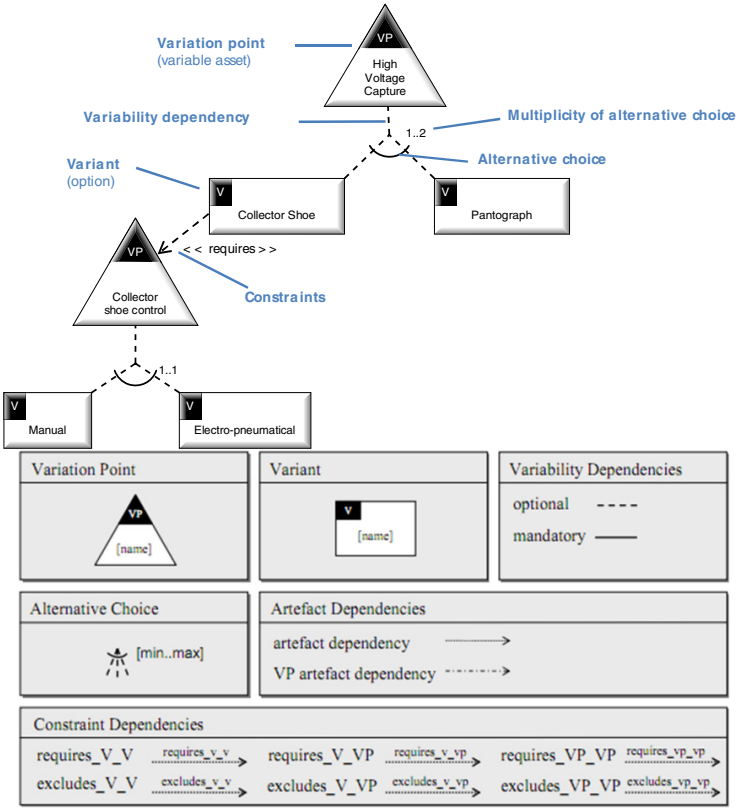


Fig. 8 Basic OVM notation

In parallel to the several engineering models, a variability model and the artefact dependencies (i.e. the links between engineering artefacts objects and variation points or variants) have to be created. The final picture of a product family model in which engineering artefacts are described by models (or have a software representation) that can be easily reused to generate a specific product *via* an instantiation process (second and third generation product line engineering) is shown in Figure 9 below.

The product family variability model includes all the possible variation points and variants of the family, at all the possible system decomposition level (system, subsystem, components, etc.) and in all the different engineering domains. A key step of this approach is to create exhaustive and consistent artefact dependencies through multiple models, while respecting the inclusion and exclusion constraints between variants, as well as the relations between the objects as defined by the global traceability meta-model shown in Figure 6.

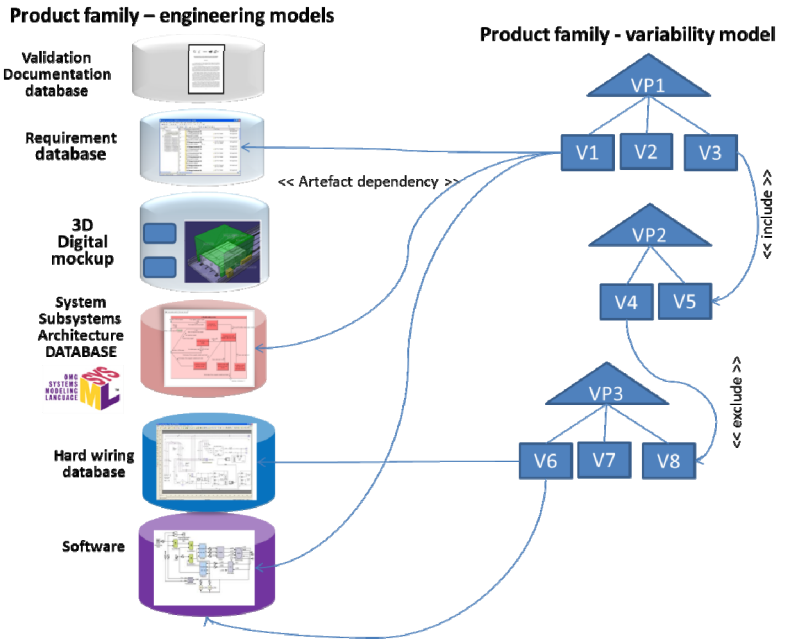


Fig. 9 Overall picture of a product family model

6 Application of a 2nd Generation Reuse Strategy to a Metro Platform

Present in many cities for over a century, metros have undergone profound transformations to meet the needs of great cities and their inhabitants. It is one of the most effective answers to road congestion and the resulting air pollution in cities. Occupying limited ground space, it permits the transportation of large numbers of passengers while limiting energy consumption.

The Metropolis product family is the ecological, intermodal and high capacity Alstom solution. It was created to meet following key requirements:

- Efficiency - the use of advanced technology optimises the performance of the traction, bogie and braking systems, as well as door opening mechanism
- Flexibility - the length and width of train sets, the number of cars and interior layouts are all modular to match all requirements,
- Reliability - outstanding communication systems facilitate maintenance and optimise safety.
- Proven standardized components, making access to equipment and maintenance easier
- Customized design for each city, offering differentiation and personalization.

- Higher capacity, combined with remarkable on-board comfort (on-board passenger information and communication system,…)
- A choice between automated driving mode and driver mode with optimized cabin ergonomics and visibility.

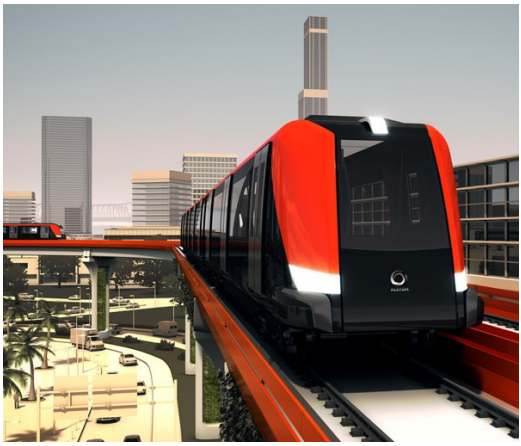


Fig. 10 One of the *Metropolis* instantiated products

For the product family described above, it has been decided to implement the second generation product line engineering approach having the requirements, the system use cases, the system/subsystem architectures (functional and constructional) and the global variability defined in models.

All the requirements are stored in a specific database which is synchronized with the SyML operational analysis and system/subsystem architectures database. This database contains the system use cases, the system functional and constructional architecture and the subsystems functional and constructional architecture. The synchronization between the two databases allows sending requirements from the requirements database to the SysML database, with the purpose of defining their satisfaction by the system or subsystem architecture elements (functions, constructional elements, interfaces, flows, etc.). Figure 11 synthesizes the architecture of our MBSE and PLE (or "MBPLSE") environment.

The variability model has been created using Atego Modeller 8.0 because it supports the OVM language and it also provides the SysML modelling capabilities that we use to model our system architectures. The formalization of the engineering artefact dependencies (requirements, use cases and architectures elements) to the elements of the variability model has also been performed completely in the Atego Modeller 8.0 tool. Using the synchronization mechanism from Atego Modeller to the DOORS database, it has been possible to import the information concerning which variant a requirements is linked to into the requirement database. This functionality is very important for the instantiation process which will be explained hereafter.

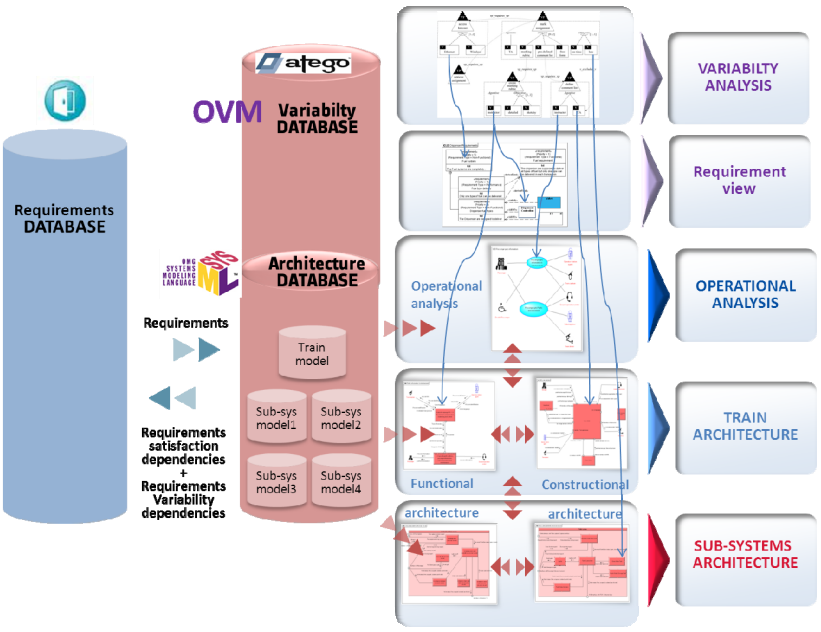


Fig. 11 Overall architecture of ALSTOM MBSE environment for the new metro product family

The second generation product line engineering uses the gap analysis process to identify which part of the product family engineering artefacts have to be instantiated to a product which will fully or partially fulfill the request for proposal. More precisely, the gap analysis is performed between the requirements and needs expressed in the request for proposal and the requirements of the 150% product family database. This database contains the description of the key requirements (performances or characteristics) of the product family, each requirement being potentially linked to one or more variants.

Once the gap analysis is completed, the set of requirements in the product family database that covers (fully or more often partially) the RFP requirements determines a preliminary partial combination of variants to be chosen in the product family to satisfy them. This combination is then verified in the variability model of the Atego Modeller 8.0 tool for consistency and exhaustiveness (exclusion and inclusion links between variation points and between variants) through specific tool functionalities. After verification, the combination of variants is used to instantiate the requirements and the architecture product family databases to produce a specific product database for the project that will develop the system. This is done by "filtering" the common elements plus the variant elements that are linked to the chosen variants. Finally, all the RFP uncovered requirements are added to the product instantiated requirement database and specific architecture solutions will be then identified and added to answer to those requirements. Figure 12 summarizes the gap analysis and instantiation process.

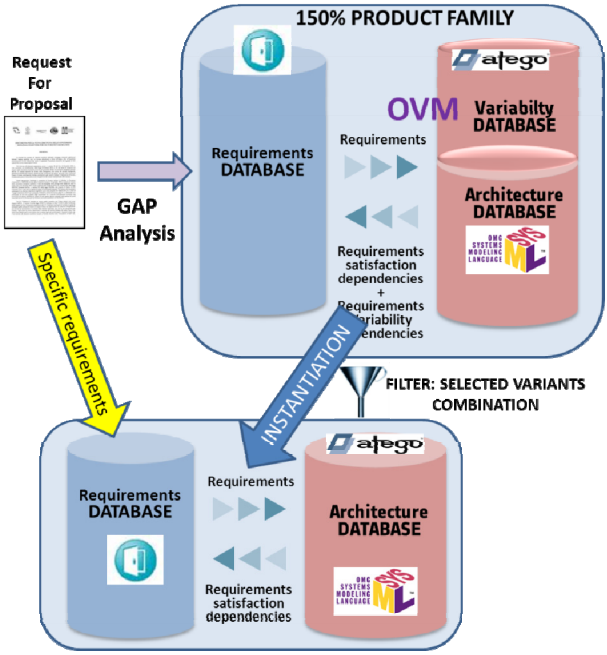


Fig. 12 Gap analysis and instantiation process overview

After the instantiation process is completed, the product family and the instantiated product will live different lives in terms of change and configuration management, but they will be anyway linked through synchronization points or milestones in which all the modifications occurred at product family level or in the several instantiated products can be shared and eventually implemented, as illustrated on Figure 13.

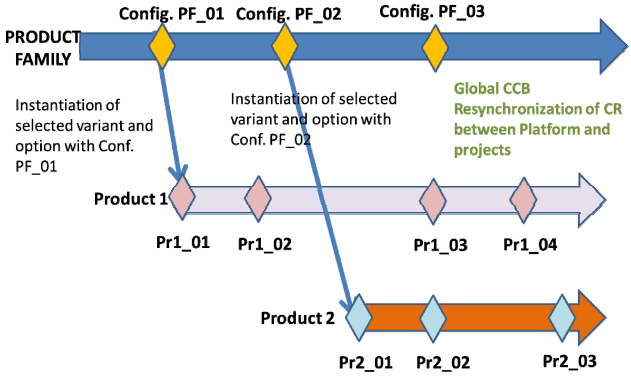


Fig. 13 Product family and instantiated product synchronization

7 Conclusions and Way Forward

In this paper, we presented a model-based product line engineering approach as well as some details on current process to instantiate products for specific projects from a product line family of systems at ALSTOM rolling stock. To summarize the current state of PLE at ALSTOM Rolling Stock, we believe that we have a good grasp on the PLE problem and a good understanding on how to implement it in a practical way. The first real application of PLE concerns the elaboration of a Metro product family and its instantiation for a current project.

Our first estimations of the benefits introduced to by this new approach yield a reduction on fixed engineering costs of about 50% (during the specification phase) in an 80% carry-over scheme. Most of the variability sources in our product family belong to the system (train) and train control / SW levels. Other than different options on technical solutions (which depend mostly on the solutions proposed by providers), little variability has been identified elsewhere at sub-system level.

There are still some aspects that need to be covered in our implementation. The next steps that we have already identified are the constitution of an asset repository and the elaboration of "modular" assets (this implies, for instance, the modification of our modelling practices). At a later stage, we will also start to specify and develop reuse efficiency measurements, which will be useful for supporting the decision making process for our projects. Finally, in order to benefit from all the potential benefits of PLE, a deeper reflection should be carried out at enterprise level to move closer to a "Full Product Line" for Rolling Stock material. The idea is to exploit all the potential of commonality between products (or between our current product line families) in order to have common engineering assets (for instance, for access door systems)shared among tram, metro and tram trains, or among sub-urban, regional and high-speed trains.

References

1. Le Put, A. (ed.): AFIS, L'Ingénierie Système d'une Ligne de Produits. AFIS (2013)
2. Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., Wsowski, A.: Cool features and tough decisions: a comparison of variability modeling approaches. In: Proceedings of the 6th International Workshop on Variability Modeling of Software-Intensive Systems (2012)
3. Davis, T.: The reuse capability model: a basis for improving an organization's reuse capability. In: Selected Papers from the Second International Workshop on Software Reusability. Proceedings Advances in Software Reuse (1993)
4. Doufene, A., ChaléGóngora, H.G., Krob, D.: Sharing the Total Cost of Ownership of Electric Vehicles: A Study on the Application of Game Theory. In: 23rd Annual INCOSE International Symposium, Philadelphia, PA. INCOSE, San Diego (2013)

5. Doufene, A., ChaléGóngora, H.G., Dauron, A., Krob, D.: Model-Based operational analysis for complex systems - A case study for electric vehicles. Accepted for presentation in the 24th Annual INCOSE International Symposium, Las Vegas, NV (2014)
6. Estefan, J.A.: Survey of model-based systems engineering (MBSE) methodologies. INCOSE MBSE Focus Group, Rev. A (May 25, 2007)
7. Flores, R., Krueger, C., Clements, P.: Second Generation Product Line Engineering – A case study at General Motors. In: Capilla, R., Bosch, J., Kang, K. (eds.) Systems and Software Variability Management –Concepts Tools and Experiences, pp. 317–341. Springer (2013)
8. Fortune, J., Valerdi, R.: Towards a Framework for Systems Engineering Reuse. In: INCOSE 2011 International Symposium, Denver, CO. INCOSE, San Diego (2011)
9. Friedenthal, S., Moore, A., Steiner, R.: A practical guide to SysML - The Systems Modeling Language. Morgan Kaufmann (2008)
10. Grady, J.O.: Universal Architecture Description Framework. Systems Engineering 12(2) (2009)
11. INCOSE Vision 2020
12. OMG Common Variability Language (CVL), OMG Revised submission, OMG (2012)
13. OMG Reusable Asset Specification (RAS), Available Specification Version 2.2, OMG (2005)
14. Pohl, K., Böckle, G., Van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag New York, Inc., Secaucus (2005)
15. Van Gaasbeek, J.R.: Model-Based System Engineering (MBSE). In: Presented in the INCOSE L.A. Mini-Conference. INCOSE (2010)
16. Wymore, A.W.: Model-Based Systems Engineering. CRC Press, Boca Raton (1993)
17. Wymore, A.W., Bahill, A.T.: When can we safely reuse systems, upgrade systems or use COTS components. Journal of Systems Engineering 3(2), 82–95 (2000)
18. Krob, D.: Eléments d'architecture des systèmes complexes. In: Appriou, A. (ed.) Gestion de la Complexité et de l'information Dans les Grands Systèmes Critiques. CNRS (2009)

An Overview of Multimodal Transport Design and Challenges Underlined by a Carsharing Case Study

Aurélien Carlier, Fabien Tschirhart, Frédéric Da Silva, François Stephan, Olivier Thoni, Alix Munier-Kordon, Manel Abid, Lionel Scremin, and Ludovic Couturier

Abstract. This paper covers some of the main aspects of multimodal transportation design (fundamental elements, topology and some existing approaches) before introducing the numerous and various challenges it poses, as a system of systems: legal and business aspects, demand prediction and multimodal planning and supervision. As an illustration, it then highlights some challenges specific to carsharing such as coping with supply, performance and patterns of the existing multimodal transportation system. Finally, using graph theory and mathematical programming, the paper studies a theoretical model to design and to optimize such a carsharing system.

Keywords: multimodal transport, carsharing, operational research, graph theory, system of systems.

Aurélien Carlier · Fabien Tschirhart · Frédéric Da Silva · François Stephan
Technological Research Institute SystemX, Palaiseau, France
e-mail: {aurelien.carlier,fabien.tschirhart,
frederic.dasilva,francois.stephan}@irt-systemx.fr

Aurélien Carlier
Renault SAS, Technocentre, Guyancourt, France
e-mail: aurelien.carlier@renault.com

Olivier Thoni
Artelys, Paris, France
e-mail: olivier.thoni@artelys.com

Aurélien Carlier · Alix Munier-Kordon
Université Pierre et Marie Curie, Paris, France
e-mail: {aurelien.carlier,alix.munier}@lip6.fr

Manel Abid · Lionel Scremin
Alstom Transport, Saint-Ouen, France
e-mail: {lionel.scremin,manel.abid}@transport.alstom.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_18

1 Introduction

In the broadest sense, multimodal transport refers to the transportation of goods or passengers performed via a transition between two different transport modes (*e.g.* rail and road). In light of the growing need for individual mobility, multimodal transportation evolves as a combination of numerous transport modes from collectives' means to individual vehicles. Both conflicting undermined limits in terms of capacity, performance and accessibility. The challenge of multimodal transportation system engineering lies in the optimization and the interoperability of intermodal passenger transport through the appropriate modelling and simulation.
Despite the existence of proper and efficient tools to handle each transport mode, the aggregated behavior of a multimodal system cannot be trivially deduced from the separated behavior of every single component. As such, we may consider multimodal transport as a system of systems, composed by a large amount of different entities, where each one has its own behavior and evolution rules, showing emergent properties.

As there is no coordination either between entities or through a common nexus, theses emergent properties cannot be considered as the result of a huge system's central planning. While its dynamics may be very complicated and include phase transition behavior (*e.g* the different flow regimes of car traffic), as its entities are hierarchically structured in a modular architecture with reproducible and programmable patterns [1][2], a multimodal transport cannot be considered as a random or a chaotic system. Hence, it fully justifies the necessity of new and more adapted models and tools to deal with this complexity of a new kind with a holistic approach. After a short introduction to multimodal transportation we will outline its main challenges: the governance organization, modelling the travel demand, following a sustainable business model, planning and supervising the network. As an illustrative purpose, we shall then discuss the case study of carsharing, which, as an emergent mode, has to consider all the different aspects of a multimodal transportation system.

2 Multimodal Transportation Design

2.1 Definition

In our study, we consider a large transport network that provides to people both public (*i.e* metro, tram, train, carsharing…) and private (*i.e* car, bicycle, walk…) transport modes as well as their corresponding operators and systems. [3] considers transportation systems as a network made out of routes or terminals : routes are simply links between two nodes, which are, as terminals, the contact or exchange points where it is possible for people to either switch to another mode, to enter the system or to leave it. On the basis of those elements, we may identify three main elements of a multimodal transportation system: travelers, transport modes, operators.

A multimodal transportation system is simply a set of transport modes which travelers may use to reach their destination from their origin. Fig. 1 shows an example of a multimodal trip in comparison with two mono-modal trips.

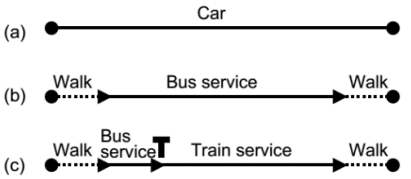


Fig. 1 (a) and (b) are mono-modal trips, and (c) is multimodal trip where T denotes the transfer point from the bus mode to the train mode.

As they are an unavoidable step for travelers to switch from one mode to another, there is no possible multimodal system without transfers. A transfer node may be used to change modes (car to train) or services (tram to train). Since multimodal transport is strongly related to transport services, in the context of public transport the term mode will be usually related to service modes. Finally, the role of walking is essential to almost any trip within a multimodal system as no transfer is possible with no walking, in addition to the necessity to walk to leave or to get into the network.

2.2 Topology and Design

When designing a transport network, two categories are considered:

- the transport service network, such as the bus or the train service network,
- the physical network, such as the road or the railroad network.

If there is a clear evidence that a bus service network is based on a road network, from the traveler's point of view, however, it will only matter whether a bus can be used for his trip. When using a transport service, any constraint that could arise from the physical network are incorporated into the transport service. However, the physical network will often determine whether a private mode is feasible for a specific trip.

The most common approach that is being used to describe a transport network is to define the network as a set of nodes connected with a set of links. Usually, a subset of nodes is used to identify access (entries and exits) and crossing nodes. Representation may include public transport lines that is a set of connected links and their nodes, and associated frequencies. It is especially suited for transportation modelling.

2.3 Some Modelling Approaches

It appears no models or algorithms are available or suitable for large-scale transport systems that simultaneously consider private and public transport networks.

Research in traffic theory encompasses an interesting set of models that can be classified, from a physical point of view, into road and rail, and from a functional point of view, into private and public modes.

Regarding road traffic flow models, there are three different descriptions of the vehicular dynamics: microscopic models, cellular automata and macroscopic models. The major advantages of macroscopic models are their tractable mathematical structure and their low number of parameters, making them faster to compute. Because of this and due to the dimensions of transport networks, macroscopic models are generally preferred over the other two. Among them, one distinguishes first-order models, second-order models, and multi-class models.

One of the most important traffic flow model is the LWR (Lighthill-Whitham-Richards) model [4], built on the analogy between road traffic flow and hydrodynamics. Thus, it does not take into account the driver attributes, while this is the case with second-order models like ARZ (Aw-Rascle-Zhang) [5] or the models of Lebacque et al. [6]. These models comprise number of elements specific to drivers such as destination, type of vehicle and behavioral attributes.

Multi-class models are from the most recent traffic flow macroscopic models. They follow the apparition in the other approaches of multi-class models. One distinguishes different classes regarding the driving behavior (desired velocity, driving style), the vehicle properties (essentially the length) and the route (destination). One can find a significant number of multi-class models [7] using different relations and algorithms.

Due to the considerably smaller number of vehicles on a railroad traffic network (in comparison to road traffic), most of models for railway traffic are microscopic. As the standard, Moving-Block systems [8] divides the railway into multiple areas, each being under the control of a computer. While operating, each train is continuously connected to the latter, so that it knows the location of every train at all time. Then, it transmits to each train the required braking curve to avoid a collision even if the leading train comes to sudden halt. Known as Pure Moving-Block (PMB), this scheme of moving-block gives the best performance and is the basis of all currently implemented systems.

3 Challenges

3.1 Governance

Today, Organizing Authorities of Urban Transports skills for the organization of transport services are, at the request of individuals, limited to regular transport. For the sale of enforcing a consistent and sustainable mobility policy, transport authorities need to extend their skills in areas affecting travel policies such as shared automotive uses and non-motorized transport modes, freight transport in the city, as well as the regulation of traffic and parking. That's why the French government undertook a reform (Law No. 2014-58 of 27 January 2014) to transform Organizing Authorities of Urban Transports into Organizing Authorities of Urban Mobility.

3.2 Demand Prediction

Travel demand modelling is one of the major building blocks for the study of the transport process. Its core objective is to produce relevant information on the potential impact of new transport infrastructures or policies on travel demand. Such information is pivotal for assessing the benefits of such projects and policy measures and to estimate their possible environmental impacts.

The fundamental approach for modelling the multimodal travel demand is the so called **4-step models** [9] [10] or sequence of models. The steps are: Trip generation, Trip distribution, Mode choice and Route assignment. Travel demand modelling refers to the first 3 steps, the last one being related to the field of traffic modelling.

4-step models are the dominant framework for operational transportation planning and policy analysis, insofar as they perform reasonably well in representing and forecasting aggregate travel demand. However, when the problems under study become more disaggregated, they may be less relevant and this shortcoming has led to the development of activity-based models. As opposed to the 4-step approach, **activity-based models** include a consistent representation of time, a detailed representation of persons and households, time-dependent routing, and micro-simulation of travel demand and traffic. This type of approach allows for a more realistic modelling of travel decisions. It provides an improved capability to model non-work and non-peak travel, to move beyond traditional explanatory zonal variables and to deal with matters like trip chaining, car sharing or links between household members.

3.3 Planning

Planning process can be classified into four steps (see [11] for an example in traffic train context):

- Strategic: crew planning (2-5 years), rolling stock acquisition, etc.
- Tactical: rolling stock scheduling (1 per year), train scheduling (1 per year), etc.
- Operational: crew scheduling (4-6 per year), timetabling (4-6 per year), etc.
- Short-term: crew scheduling (daily), timetabling (4-6 per year), etc.

One of the biggest challenges in transport planning lies in having an overall consistency between each mode. Currently, each mode has its own planning. For example, in case of perturbation, the report from one mode to another can generate heavy problems on this one. So it's really necessary to have predictive modelling capacities to be able to anticipate problems and to test strategies that will be directly exploited in short-term planning.

3.4 Business Model

The ecosystem of urban transports is composed by a lot of public and private actors. A simple enumeration of these different actors is not enough to describe the complexity of a transportation system. Indeed, relations between these actors imply other considerations (for example: juridical or economic aspects).

In fact every business actor optimizes its own business. But for a global optimization of the multimodal transport system, an optimized combination of transport modes is requested. That's why the real challenge lies in having a generic model that describes the complete framework of the urban transports ecosystem. Thanks to this kind of model, it's possible to simulate economic transfers between actors and to optimize and compare different scenarios of a multimodal transport system.

3.5 Multimodal Supervision

Multimodal supervision raises many challenges because it involves the monitoring and the control of several modes that differ in various ways including their availability, density, costs, etc. Indeed, these modes are not perfect substitutes, each one is more appropriate for specific users and uses. Hence, the real-time coordination and synchronization of system as a whole may easily become a very complicated task. From the end users point of view, the challenge is to guarantee mobility with an expected quality of service, whatever the conditions. From the legal point of view, the challenge is to make the stakeholders cooperate on a contractual basis. From a business point of view, the big challenge is to define operational strategies and solutions with global optimization criteria.

4 Carsharing, a New Transportation Mode at the Crossroads of All Challenges

This part will deal with a specific use case of a carsharing system. It's an interesting new emergent transport mode that handles all the previously described challenges.

4.1 Presentation, Urban Impact and Challenges

Since the mid-twentieth century, the greater accessibility to the private car in industrialized countries has significantly improved the people mobility in urban areas. While this new mode of transportation greatly helped societies realize their aspiration for growth and prosperity, it also resulted in serious negative externalities: pollution, excessive consumption of energy and time due to congestions problems, etc. To control, manage and deal with those problems, a lot of efforts are made to found alternative solutions [12]. One of them is carsharing system which involves a small to medium fleet of vehicles, available at several stations

distributed over a given geographic area, to be used by a relatively large group of members [13]. Although the first identified carsharing system appeared around the mid-twentieth century, such systems became popular worldwide since the early 1990's. They represent a real alternative to private car and release the user from constraints related to individual property since the carsharing company is in charge of insurance, maintenance, fuel (or electricity), taxes, depreciation, etc. Different studies (see for example [14, 15]) have evaluated that for a user driving less than 10,000 kilometers per year (as much as 15,000 km/y), it also could be a real alternative to private car, in a financial way, depending on local costs. Since then, we can found over the world two different types of carsharing systems.

Historically, the first one requires users to return vehicles to the station they were picked up. These are called "round-trip" carsharing systems and are the most common. They are simple to manage since the demand for each station is enough to dimensioning the station. The user behavior in such systems is mainly oriented to leisure and household shopping purpose ([16, 17]). The second one, called "one-way" carsharing system, is much more flexible for the user since it allows the latter to pick up a vehicle from a station and return it in a different one, which can be different from the origin. Unfortunately, this greater flexibility comes with hard operational problems due to the uneven nature of the trip pattern in urban areas. However, it is worth mentioning that despite these difficulties for the operator, one-way system captures more trips than the alternative system thanks to this flexibility which is, as showed in [18], a critical factor to joining a carsharing scheme. In the last decade, several authors have showed that these systems have a positive impact on urban mobility, mainly because of higher utilization rate than private vehicle ([14, 19]). Indeed, shared vehicles spend more time on the road and less time parked (which represent for a private car almost 95% of its total use time, as mentioned in [20]), thereby decreasing parking requirements in dense areas [12] and reducing the average number of vehicles per household ([21, 22]). It also decreases the total number of vehicles on the road, since one vehicle can be driven by several users and thus improving the traffic fluidity. Furthermore, it's now recognized that carsharing systems have positive environmental effects. It reduces greenhouse gas (GHG), CO_2 emissions ([23, 24]) and provides noise reduction since electric cars are quitter than thermal ones. The reduction of parking demand can also be used to reallocate the land for additional green spaces, new mixed-use development, or other community needs [25].

Thus, carsharing systems seems to be a very attractive and profitable solution for transportation issues, improving on the one hand the global transportation system efficiency in dense areas and bringing on the other hand a significantly ecological impact in the urban environment. As mentioned in [26] and because of their relatively recent emergence, they must be devised taking into account the specificities of the whole multi-modal transportation system: the existing supply, its operational performance, the inter-relations between existing modes, the economic associated models, the travel patterns and behaviors of the travelers, etc. This is a real challenge, not only because of the modelling complexity of such systems, but also due to the collect and the estimation of realistic data concerning

a lot of different aspects, from the most strategic to the operational. It's now known that a lot of travels through the transportation system are using more than one mode and any user of a given mode can almost come from any other existing mode. Then it turns out that in order to tackle the dimensioning of a carsharing system, it's crucial to be able to describe and capture the amount of demand that switches over modes, taking into account departures/destinations, existing multimodal infrastructure and time.

4.2 Case-Study: Dimensioning Carsharing Fleet within a Multimodal Transportation System

The problem discussed here consists in finding the optimal configuration, in terms of stations size and fleet size, of a set of possible carsharing stations when demand and travel times are given over time. As several studies (see for instance [27–29]), we will use graph theory and mathematical programming to tackle this problem, attempting to integrate their results and recommendations. A lot of them integrate relocation operations between stations that, as showed in [30, 31], allow the operational system to reach an efficiency level that cannot be achieved otherwise. This characteristic seems then necessary since we are interesting in the best system performance although it comes with many challenges, especially in terms of complexity and computing time.

However, we want to introduce here some differences with previous research. Most of them tried to maximize the carsharing operator revenue, whereas we will focus on the system efficiency in terms of number of demand it can handle. In our view, it could also be very interesting to design the carsharing system taking into account multiple objective optimization. We selected three criteria: number of satisfied demand, number of relocation operations and number of vehicles.

The main idea is to consider a Time Expanded Graph (TEG), introduced in [32], where the nodes represent the stations over a given set of discrete time-steps $\mathcal{H} = \{0, \dots, T\}$ and arcs symbolize the "movements" of vehicles. These are defined through three distinct sets. A first one called E_1 represents the vehicles parked in stations between two consecutive time-steps. In that case, arcs could be viewed as a stock rather than a movement. The capacity on these arcs are fixed to the maximum station size (number of parking slots). Then, a second set E_2 will capture the demand of vehicles from a station to another at a given time-step. This time, the capacity is set to the number of vehicles required to that specific demand. Finally, a third set E_3 will represent the possible relocation operations of vehicles between each pairs of stations for every time-step. This last set admits infinite capacities on its arcs. Arcs of E_2 and E_3 are defined such that the time step of each destination node correspond to the departure time-step from origin station plus the time that a passenger would make to do the trip, including penalties depending on the travel context, as congestion for instance. Fig. 2 gives an example of these sets.

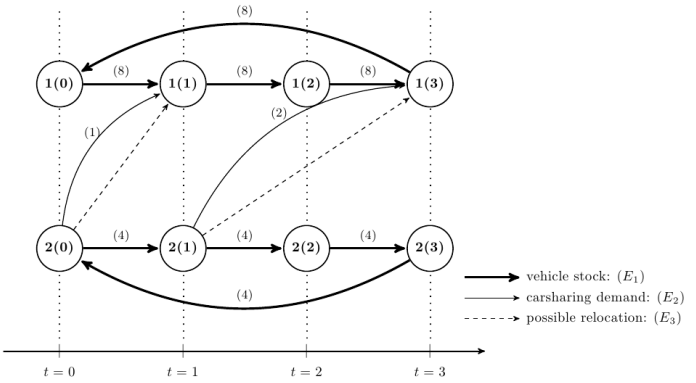


Fig. 2 Time Expanded Graph example for the carsharing fleet dimensioning problem with two stations and four time-steps

Fig. 2 *r*epresents the time-states of two stations named "1" and "2" placed horizontally over four time-steps. Capacities are put in brackets over each arcs, except those of relocation operations. The arc of demand which starts from node "$2(0)$" to node "$1(1)$" means that a possible passenger wish borrow a vehicle from station "2" to station "1" at time $t = 0$. The same reasoning stands for all the arcs of demand and relocation. Let's also note the cyclical aspect of the resulting graph. All the arrival time-steps used for the arc definition are calculated modulo T such that the time space $\mathcal{H}$ must represent a classical and homogenous time situation, as an average week day for example.

Thus, the resulting problem consists in finding a maximal flow of vehicles transiting through the graph, maximizing the sum over the arcs of demand and respecting the classical constraints of flows problems: capacity constrains over each arcs and flow conservation over each nodes. Every cut between two distinct time-steps will give the number of vehicles used in the system.

Using a random generator, which produce such time-expended graph with realistic data, we started looking problem solutions with the open-source linear programming solver GLPK [33]. A good manner to study multi-objectives optimization is to use Pareto frontiers. Thus, we present thereafter a 3-dimentionnal Pareto frontier giving optimal demand for different values of two other objectives (total number of relocations operations and total number of vehicles transiting through the system; see Fig. 3). The later both objectives are intended to be minimized, while we are interesting into the greater number of satisfied demand. The instance generated is a simple case study: 50 demands over 3 stations during 144 time-steps (an entire day with a time-step each 10 minutes). The travel times take into account two key moments of a classical week day in our urban area: morning and evening rushes. For those time slots, penalties are integrated in the travel time computing process in order to be more realistic.

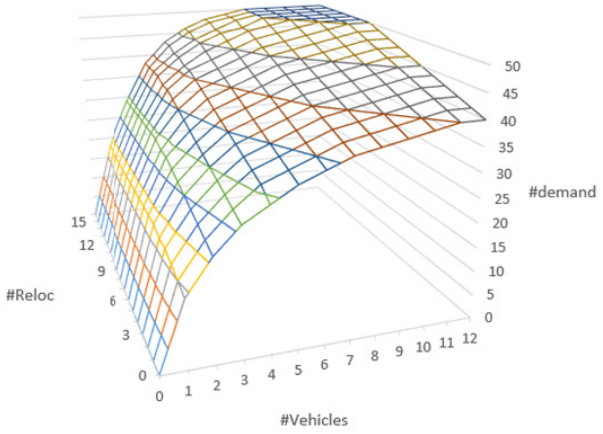


Fig. 3 3-dimentionnal Pareto frontier of a case-study generated instance

As expected, the more relocation operations or vehicles in the system, the higher the satisfied demand. The different colored strata provide a first insight of the front looks for different range of demand values. Number of vehicles allowed during the optimization also seems to have more impact on the demand than the number of relocations, while appearing to be correlated. Further research will clarify such link between the two objectives as well as testing the optimization computing on bigger instances with real data. Actually, first numerical results on those topics are currently pending publication. Also, future research shall look forward to investigate the system's behavior when relocation operations are done with autonomous cars.

5 Conclusion

From the previous study, it appears that dimensioning and optimizing a car sharing system cannot be trivially deduced from the desired behavior of every single vehicle of its fleet. Indeed, such a system has to handle many different factors and issues specific to multimodal transportation: from both the designing and the topology; but also to the various challenges we previously outlined: in terms of governance and demand prediction, planning, business model and supervision. Also, in the case of multimodal transportation, finding optimal solutions for a specific mode is probably not be possible as long as it is sidelined [26]. Consequently, considering multimodal transportation as a system of systems will allow the simultaneous integration and merging of multiple transport modes. This is the research scope of the MIC (Modelling-Interoperability-Communication) project within Technological Research Institute SystemX which is looking forward to enabling the sizing, the positioning, the optimization and the supervision of multimodal mobility systems, using simulation models.

Acknowledgement. This research work has been carried out under the leadership of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir". Special thanks are formulated to all contributors of the MIC project: Alstom, Artelys, The Cosmo Company, CEA, IDIT, IFSTTAR, Inria, Renault, SNCF, UPMC.

References

1. Maier, M.W.: Architecting principles for systems-of-systems. Syst. Eng. 1, 267–284 (1998), doi:10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D
2. Aiguier, M., Boulanger, F., Krob, D., Marchal, C.: Complex Systems Design & Management. In: Proc. Fourth Int. Conf. Complex Syst. Des. Man-Agement, CSDM 2013 (2013)
3. Tolley, R.S., Turton, B.J.: Transport systems, policy and planning: a geographical approach. Longman, New-York (1995)
4. Lighthill, M.J., Whitham, G.B.: On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. Proc. R. Soc. Math. Phys. Eng. Sci. 229, 317–345 (1955), doi:10.1098/rspa.1955.0089
5. Fan, S., Herty, M., Seibold, B.: Comparative model accuracy of a data-fitted generalized Aw-Rascle-Zhang model. ArXiv Prepr ArXiv13108219 (2013)
6. Lebacque, J.P., Khoshyaran, M.M.: A Variational Formulation for Higher Order Macroscopic Traffic Flow Models of the GSOM Family. Procedia - Soc. Behav. Sci. 80, 370–394 (2013), doi:10.1016/j.sbspro.2013.05.021
7. Kessels, F.L.M.: Multi-class continuum traffic flow models: analysis and simu-lation methods. TRAIL (2013)
8. Pearson, L.V.: Moving block railway signalling. PhD Thesis, Loughborough University (1973)
9. Bonnel, P.: Prévoir la demande de transport. Presses de l'École Nationale des Ponts et Chaussées (2004)
10. de Dios Ortezar, J., Willumsen, L.G.: Modelling Transport. John Wiley & Sons (2011)
11. Acuña-Agost, R.: Mathematical modeling and methods for rescheduling trains under disrupted operations. Université d'Avignon (2009)
12. Mitchell, W.J.: Reinventing the automobile: Personal urban mobility for the 21st century. MIT Press (2010)
13. Shaheen, S.A., Sperling, D., Wagner, C.: A Short History of Carsharing in the 90's (1999)
14. Litman, T.: Evaluating carsharing benefits. Transp. Res. Rec. J. Transp. Res. Board 1702, 31–35 (2000)
15. Prettenthaler, F.E., Steininger, K.W.: From ownership to service use lifestyle: the potential of car sharing. Ecol. Econ. 28, 443–453 (1999)
16. Barth, M., Shaheen, S.A.: Shared-use vehicle systems: Framework for classi-fying carsharing, station cars, and combined approaches. Transp. Res. Rec. J. Transp. Res. Board 1791, 105–112 (2002)
17. Costain, C., Ardron, C., Habib, K.N.: Synopsis of users' behaviour of a carsharing program: A case study in Toronto. Transp. Res. Part Policy Pract. 46, 421–434 (2012), doi:10.1016/j.tra.2011.11.005

18. Efthymiou, D., Antoniou, C., Waddell, P.: Which Factors Affect the Willing-ness to Join Vehicle Sharing Systems? Evidence from Young Greek Drivers (2012)
19. Schuster, T., Byrne, J., Corbett, J., Schreuder, Y.: Assessing the Potential Extent of Carsharing: A New Method and Its Implications. Transp. Res. Rec. J. Transp. Res. Board 1927, 174–181 (2005), doi:10.3141/1927-20
20. Transflash - Bulletin d'information des déplacements urbains départementaux et régionaux. Certu (2013)
21. Martin, E., Shaheen, S.A., Lidicker, J.: Impact of carsharing on household vehi-cle holdings. Transp. Res. Rec. J. Transp. Res. Board 2143, 150–158 (2010)
22. Ter Schure, J., Napolitan, F., Hutchinson, R.: Cumulative impacts of carsharing and unbundled parking on vehicle ownership and mode choice. Transp. Res. Rec. J. Transp. Res. Board 2319, 96–104 (2012)
23. Martin, E.W., Shaheen, S.A.: Greenhouse gas emission impacts of carsharing in North America. IEEE Trans. on Intell. Transp. Syst. 12, 1074–1086 (2011)
24. Firnkorn, J., Müller, M.: What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm. Ecol. Econ. 70, 1519–1528 (2011)
25. Cohen, A.P., Shaheen, S., McKenzie, R.: Carsharing: A Guide for Local Planners (2008)
26. Jorge, D., Correia, G.: Carsharing systems demand estimation and defined operations: a literature review. EJTIR 13, 201–220 (2013)
27. Kek, A.G., Cheu, R.L., Chor, M.L.: Relocation simulation model for multiple-station shared-use vehicle systems. Transp. Res. Rec. J. Transp. Res. Board 1986, 81–88 (2006)
28. Correia, G.H., de, A., Antunes, A.P.: Optimization approach to depot location and trip selection in one-way carsharing systems. Transp. Res. Part E Logist. Transp. Rev. 48, 233–247 (2012), doi:10.1016/j.tre.2011.06.003
29. Jorge, D., Correia, G., Barnhart, C.: Testing the Validity of the MIP Approach for Locating Carsharing Stations in One-way Systems. Procedia - Soc. Behav. Sci. 54, 138–148 (2012), doi:10.1016/j.sbspro.2012.09.733
30. Wang, H., Cheu, R., Lee, D.-H.: Dynamic Relocating Vehicle Resources Using a Microscopic Traffic Simulation Model for Carsharing Services. In: Third Int. Jt. Conf. on Comput. Sci. Optim. CSO 2010, pp. 108–111 (2010)
31. Nair, R., Miller-Hooks, E.: Fleet management for vehicle sharing operations. Transp. Sci. 45, 524–540 (2011)
32. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: theory, algorithms, and applications. Prentice Hall, Englewood Cliffs (1993)
33. GLPK - GNU Linear Programming Kit, http://www.gnu.org/software/glpk/

Timed Symbolic Testing Framework for Executable Models Using High-Level Scenarios*

Mathilde Arnaud, Boutheina Bannour, Arnaud Cuccuru,
Christophe Gaston, Sebastien Gerard, and Arnault Lapitre

Abstract. Refining high-level system scenarios into executable models is often not automatic and subject to implementation choices. We develop techniques and tools combining different modes of simulation in order to assess automatically the correctness of executable fUML activities with respect to system scenarios specified as UML MARTE sequence diagrams. In this paper, we show how test data are extracted from sequence diagrams using symbolic execution and how they are used as inputs to test system activities in the standardized fUML virtual machine.

1 Introduction

Over time, software systems tend to be used in highly critical scenarios in a variety of areas. Examples include advanced driver assistance systems, autopilots in aircraft or railway systems, etc. Such systems are often made of multiple components which are highly concurrent, and are all tied together in complex ways. Hence, standards under which large-scale software systems have to be developed are increasingly stringent and demanding regarding confidence in their quality. Formal methods are essential to achieve higher confidence levels since they allow replacing validation operations done manually

Mathilde Arnaud · Boutheina Bannour · Arnaud Cuccuru · Christophe Gaston · Sebastien Gerard · Arnault Lapitre
CEA, LIST, LISE Laboratory
Point Courrier 174, 91191, Gif-sur-Yvette, France
e-mail: {firstname.lastname}@cea.fr

* Work partially supported by the European openES project.

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_19

by automatic or semi-automatic techniques with mathematical foundations justifying their use. Unfortunately, the reference models used in validation are often large and therefore it is unclear how much confidence to award them. A top-down process which relies on refinement techniques should be used in order to shift the burden of formal analysis from detailed reference models to the correctness of refinement steps. In this paper, we do not define refinement mechanisms for transforming high level models into low level executable models. We rather consider the refinement as maintaining a correctness relation which states sufficient conditions on conforming executable models with respect to high level scenarios. The objective of the paper is to investigate a refinement approach integrated with the Unified Modeling Language (UML) which is a graphical language that can be used to design complex software systems at different levels of abstraction. A complex system can be first specified as a UML sequence diagram with timing properties using the constraint language MARTE::VSL [9]. In sequence diagrams, one may describe execution scenarios in terms of partially-ordered sequences of messages exchanged between basic interaction points (called ports) owned by components to communicate with their environment. Message descriptions may include constraints on the type of value transmitted and the time at which the message is processed. Conceptually, sequence diagrams characterize requirements on system behaviors while abstracting as much as possible internal computation flows inside components. Later in the design cycle, a more detailed model of each component behavior may be designed with UML activity diagrams. Activities in UML are flow charts built with communication and computation actions. System activities are deterministic and directly executable according to the fUML execution semantics [11]. In our approach, we develop a tooled testing process to assess automatically the conformance of the system activities with respect to sequence diagrams. We also pay particular attention to evaluating compliance with timing constraints since the system depends on those constraints being satisfied to operate correctly. Test data are generated from sequence diagrams by symbolic execution [13] which improves behavioral coverage and hence fault-detection capability of the testing process.

This paper is organized as follows. The section 2 gives an overview of our approach and tools. It also introduces an automotive system used as a running example in this paper. Section 3 describes how sequence diagrams are used in our approach to specify high-level scenarios and their symbolic treatment. Section 4 shows how activities of components in the system are designed and presents their execution semantics in the fUML virtual machine. Section 5 defines the testing process. Section 6 presents the experimental results and discusses practical issues in test coverage within the context of our models. Section 7 reviews some related works. Finally, Section 8 draws the conclusions.

2 Approach Overview

The goal of this section is twofold. First, we briefly discuss the tools involved in the approach, and then we present the approach itself which is illustrated step-by-step in Figure 1.

• *Papyrus* is the tool used for modeling UML diagrams[1]. Papyrus is a graphical editing tool for UML2 integrated with Eclipse.

• *Moka* is an eclipse plug-in which aims at providing support for model execution in the context of Papyrus [20]. *Moka* provides basic execution, debugging and logging facilities for foundational UML (fUML) [11], an executable subset of UML with precise operational semantics. We use activity diagrams that are part of fUML to model the internal behavior of components.

• `sdToTIOSTS` is an eclipse plug-in for Papyrus. It is used to translate sequence diagram models into Timed Input Output Symbolic Transition Systems (TIOSTS) [3]. TIOSTS are symbolic automata with time guards. Resulting TIOSTS can be analyzed by means of the symbolic execution tool *Diversity*.

• *Diversity* is a symbolic automatic analysis and testing tool [4]. *Diversity* uses symbolic execution techniques to compute a symbolic tree representing all the possible executions of a TIOSTS. A symbolic path represents all the timed traces that can be computed by satisfying path conditions on data and time. *Diversity* offers coverage criteria such as transition coverage. Finally, *Diversity* is coupled with sat-solvers in order to generate timed traces associated to symbolic paths and to test whether a timed trace reveals non-conformance relatively to a trace-based conformance relation called tioco [19].

Let us overview the different steps of the approach in Figure 1. The first step (1) consists in specifying system scenarios which describe the intended interactions between all components of the system. The second step (2) consists in refining the scenarios into an executable model which specifies with an activity diagram the internal behavior of each component. System scenarios as sequence diagrams are analyzed with *Diversity* in step (3). For each sequence diagram, *Diversity* computes a symbolic tree, where each path denotes a possible (symbolic) execution in the sequence diagram. Then a path is selected in step (4) relating to a specific behavior and a sequence of stimuli (as a timed trace) is extracted from it. Next in step (5), the fUML virtual machine of the tool *Moka*, being supplied with the test stimuli, is used to set up a test environment and execute the system activities. In Step (6) the system responses are collected by *Moka*. The latters are taken as inputs by the testing algorithm in *Diversity* which computes in step (7) a verdict concerning the *tioco*-conformance of execution and the coverage of the requirement. Naturally, in case of fault-detection the system activities need to be revised by the designer.

Automotive example. For the rest of the paper, we will use a running example whose structure is depicted in Figure 2. It specifies a rain-sensing wiper

[1] `www.eclipse.org/papyrus`

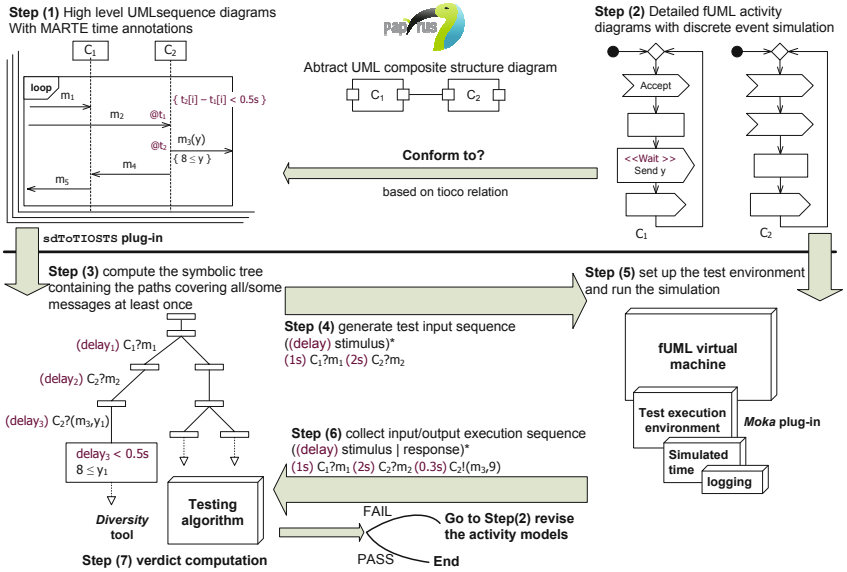


Fig. 1 Validation Process

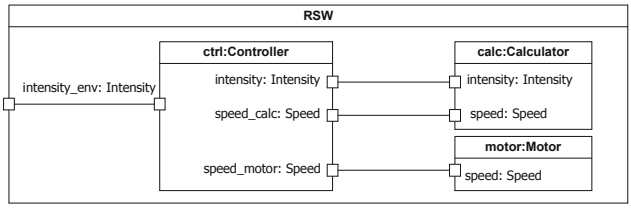


Fig. 2 RSW composite structure diagram

system in a car, denoted RSW. Three components are involved in the RSW system: a controller, a calculator and a wiper motor. These are some of the requirements that must hold for the actual implementation of the system: **(R1)** RSW adjusts the wiping speed according to the amount of rain detected; **(R2)** RSW controls automatically the adjustment of the wiper activity; and **(R3)** RSW response time is less than 0.5 seconds after detection.

3 Interaction Scenarios and Symbolic Simulation

We develop next high level system scenarios and explain how we analyze them.

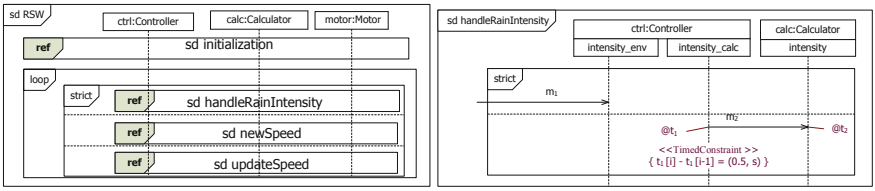


Fig. 3 RSW sequence diagram–Subdiagram representing the handling of the received rain intensity

3.1 Sequence Diagrams

The sequence diagrams given in Figures 3–5 describe the global behavior expected from the RSW components. Each of the ports is represented by a lifeline. The behavior described in the diagram is repetitive and the number of iterations is not known beforehand, which is captured by the *loop* operator. The controller receives inputs from the environment in message m_1 and sends periodical updates to the calculator about the rain intensity on channel m_2. Note that the messages are asynchronous and may thus be received at a later date than their emission. The periodicity of the updates is given by a time constraint of the form $t_1[i] - t_1[i-1] = (0.5, s)$, where i represents how many times the behavior has looped and t_1 is an array containing at index i the time value associated. The calculator then computes what speed the wiper should adopt given such intensity: this is represented in the sequence diagram by the computation of a new value for the speed variable : $new(speed)$. The calculator sends the result of this computation back to the controller over message m_3. Then there are two alternatives as captured by the *alt* operator: either the new computed value for the speed of the wiper is the same value as the previously computed speed, and in that case nothing need be done, or the new value is different. In that case the value of the stored previous speed is updated and a signal is sent to the motor with the new speed value (message m_4). In a sequence diagram, the lifelines are running asynchronously. In order to synchronize some executions, we use the *strict* operator which ensures that the behavior in the first part is finished before the behavior in the second part begins.

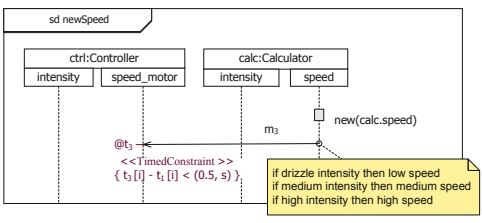


Fig. 4 Controller computing new speed

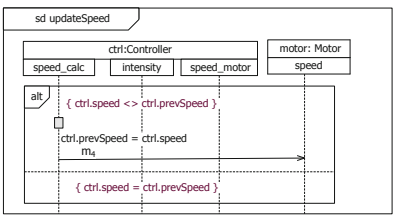


Fig. 5 Updating speed

3.2 Symbolic Execution

In order to formally reason about sequence diagrams, we provide them with formal semantics given by TIOSTS automata [3]. Symbolic execution may be carried out on such automata and thus we obtain timed traces representing possible behaviors characterized by the sequence diagram.

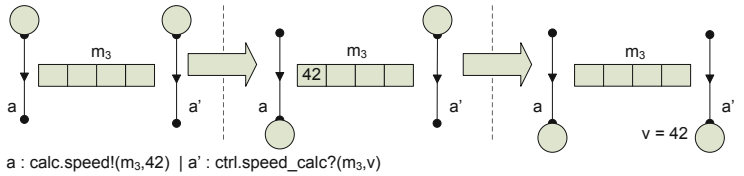


Fig. 6 Example of transitions representing an asynchronous communication

Translation into communicating TIOSTS. Let V be a set of variables and C be a set of channels. A TIOSTS over (V, C) is a triple (Q, q_0, T) where Q is a set of states, q_0 is the initial state and T is the set of transitions. Transitions are of the form $q \xrightarrow{t,\phi_t,\phi_d,act,\rho} q'$ where q, q' are states, t is an array of time instants, ϕ_t and ϕ_d are guards resp. over the time and the data constraints, ρ is a substitution over variables of V and act is one of the possible actions that can be triggered. Possible actions are: receiving a value x on channel $c \in C$ denoted by $c?x$; sending value u on channel $c \in C$ denoted by $c!u$; assigning a new value to variable x denoted by $new(x)$; and the empty action τ. We translate a sequence diagram into a set of TIOSTS: each lifeline is translated into a TIOSTS, by translating successive events into transitions with the appropriate constraints. We only detail the translation of the asynchronous communications which was not considered in [3]. An unbounded FIFO variable is associated to each channel in order to emulate the communication actions. Each time a message m is sent on a channel c by a lifeline l, it writes m on the FIFO associated to channel c. Each time a lifeline receives on channel c, it reads on the FIFO associated with c the message. This process is illustrated in Figure 6.

Symbolic tree. Reasoning with concrete input values can result in a very large, possibly infinite, number of executions of the system. We use symbolic execution techniques instead. The underlying idea is to abstract some of the values, be they data values or time values, as variables, and thus characterize classes of executions. Besides data, we define symbolic handling of TIOSTS time variables. Symbolic states allow storing information about the execution of the system that may constrain the values of the variables in *path conditions.*

A symbolic execution corresponds to a concrete one if and only if the collection of path conditions is satisfiable. For example, the path condition collected in one path of Figure 7 is made of two parts, the time path condition $PC_t = d_0 + d_1 + d_2 < 0.5s$ and the data path condition $PC_d = calc.speed_1 <>$

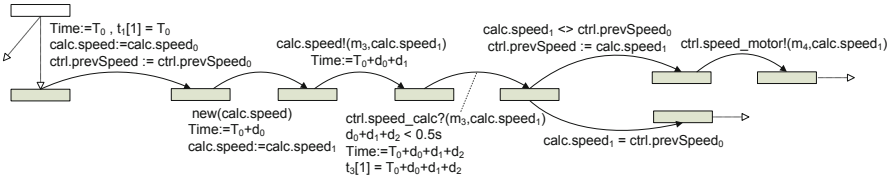


Fig. 7 Part of the symbolic tree of the RSW sequence diagram

$ctrl.prevSpeed_0$. Using solving techniques on this path condition, we can deduce concrete traces.

Timed traces of sequence diagrams. We use a set D of durations and a data model M which includes most common types. A sequence diagram SD is defined over a signature $(\mathcal{P}, \mathcal{M}sg)$ where $\mathcal{P}$ is a set of ports and $\mathcal{M}sg$ is a set of messages. The set of communication actions on port $p \in \mathcal{P}$ is $Act(p)$ of the form $I(p) \cup O(p)$, where $I(p) = \{p?(m,v) | m \in \mathcal{M}sg, v \in M\}$ and $O(p) = \{p!(m,v) | m \in \mathcal{M}sg, v \in M\} \cup \{\overline{p}!(m,v) | m \in \mathcal{M}sg, v \in M\}$: an action of the form $p!(m,v)$ corresponds to an emission of a message by p; $\overline{p}!(m,v)$ corresponds to a reception of a message sent by an internal component of the system, and $p?(m,v)$ corresponds to a reception of a message coming from the environment of SD. We define the set $I(SD)$ of all inputs (resp. outputs) in SD as $\bigcup_{p \in \mathcal{P}} I(p)$ $(\bigcup_{p \in \mathcal{P}} O(p))$. The set of all communication actions in SD is $I(SD) \cup O(SD)$, denoted $Act(SD)$. A *timed trace* of a sequence diagram SD is a word from $(Act(SD) \cup D)^*$ which respects the causal order inferred from the sequence diagram together with the timing constraints and the performed computations on inputs. The define $TTraces(SD)$ as the set of all timed traces of SD.

4 Activity Diagrams and Numeric Simulation

The objective of this section is to introduce a subset of activities that we use and discuss their underline execution semantics in the fUML virtual machine.

4.1 Activity Diagrams

Components involved in the system are refined by designing an activity diagram for each individual component. Each activity diagram specifies the communication and the computation logic. Figure 8 illustrates activities associated with the controller and the calculator of the RSW system.

Both activities specify a cyclic behavior. Let us discuss actions of the controller:

• *AcceptEventActions* (nodes *Start*, *Accept SpeedSignal*, and *Accept IntensitySignal*) specify synchronization points, where the controller waits for inputs from its environment;

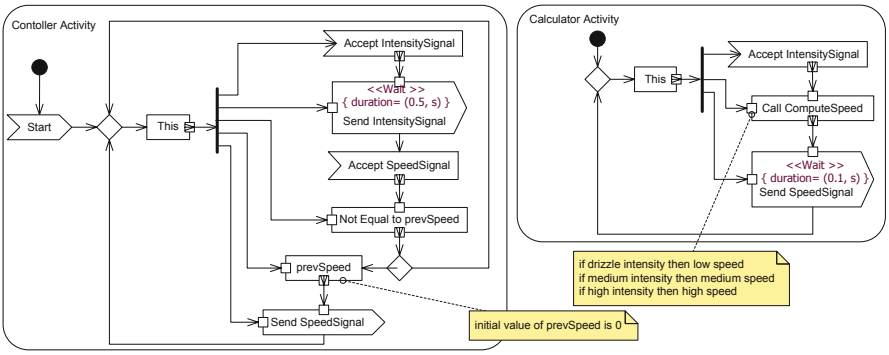


Fig. 8 RSW activity diagrams

• *SendSignalActions* (nodes *Send IntensitySignal*, and *Send SpeedSignal*) specify asynchronous communications between the Controller and its environment;
• Other actions specify computations or access to context information of the component object: *Not Equal to prevSpeed* determines if the received speed value is equal to the previous one for example.
Nodes may be annotated (using the stereotype <<Wait>>) with durations which must elapse before they are executed. E.g. *Send IntensitySignal* corresponds to sending a rain intensity value to the calculator immediately after a $0.5s$ delay.

4.2 fUML Virtual Machine and Discrete-Event Simulation

The fUML virtual machine (VM) is implemented in the Moka tool. It allows the execution of fUML activities of components structured with UML composite diagrams. A specific simulation library defining a Model of Execution (MoC) is responsible for controlling the execution and simulating extra-functional aspects such as timing features. In this paper, we use a particular MoC, *discrete-event MoC* which introduces a discrete model of time. During the traversal of the control flow of the activity, an event is triggered by the fUML VM each time a communication action is interpreted and is stored in the event queue of a scheduler. Events correspond to fUML signals, carrying a delay. This value indicates the time when this communication node will be woken up. During the execution, the event with the smallest delay is then selected, firing the communication action referenced by this event and removing it from the event queue. Once communication actions are fired, the fUML VM is in charge of propagating the values through the connector architecture conforming to [10]. Finally as glimpsed in Section 2, we need to collect executions traces in order to analyze their conformance: to that

end we integrate in the fUML VM run-time logging capabilities.Thanks to discrete-event MoC, the execution trace is enriched with durations that have elapsed between exchanged data.

5 Conformance Testing

We present in this section the *tioco* conformance relation and then, we describe in practice the conformance testing process.

5.1 Conformance Relation

In our settings, an *Activity model* $\mathbb{A}$ defined over a set of ports $\mathcal{P} = \amalg_{i \leq n} \mathcal{P}_i$, is a finite set $A_1, \ldots, A_l$ of activities defined respectively over $\mathcal{P}_1 \ldots \mathcal{P}_n$ where n is the number of components in the system. The *Activity model* is directly executable in the fUML virtual machine as explained in Section 4. The simulation history of an activity model can be mathematically characterized as a set of traces defined over the set of ports denoted $TTraces(\mathbb{A})$. We also define an *Interaction model* $\mathbb{S}$ defined over $(\mathcal{P}, \mathcal{M}sg)$ as the set of k sequence diagrams $SD_1, \ldots, SD_k$ defined respectively over $(\mathcal{P}_1, \mathcal{M}sg_1) \ldots (\mathcal{P}_k, \mathcal{M}sg_k)$ such that: for all $i \leq k$,$\mathcal{P}_i \subseteq 2^{\mathcal{P}}$; and $\mathcal{M}sg = \amalg_{i \leq k} \mathcal{M}sg_i$. That is, a sequence diagram in $\mathbb{S}$ may include only a subset of all system ports and the sets of messages are disjoint. Timed traces of an interaction model $\mathbb{S}$, denoted $TTraces(\mathbb{S})$, is the union of timed traces of all sequence diagrams in $\mathbb{S}$, $\bigcup_{i \leq k} TTraces(SD_k)$ (we define similarly the set of all input and output sets of $\mathbb{S}$ resp. $O(\mathbb{S})$ and $I(\mathbb{S})$). Let $\sigma \in TTraces(\mathbb{S})$, we define the auxiliary function $h(\mathcal{M}sg, \sigma)$ as follows: if σ is of the form $p_1 \diamondsuit (m_1, v_1) \ldots p_n \diamondsuit (m_n, v_n)$, where $\diamondsuit \in \{?, !\}$ then $h(\mathcal{M}sg, \sigma) = p_1 \diamondsuit v_1 \ldots p_n \diamondsuit v_n$; otherwise $h(\mathcal{M}sg, \epsilon) = \epsilon$. Let us consider further the following definition: Let σ_1, σ_2 be two traces respectively in $TTraces(\mathbb{S}), TTraces(\mathbb{A})$. σ_2 is not distinguishable from σ_1 w.r.t $\mathcal{M}sg$, denoted $\sigma_2 \sim_{\mathcal{M}sg} \sigma_1$, if and only if $\sigma_2 = h(\mathcal{M}sg, \sigma_1)$. We adapt in the following definition the conformance relation *tioco* [19] in order to define the correctness of an activity model $\mathbb{A}$ w.r.t an interaction model $\mathbb{S}$.

Definition 1 (tioco). Let $\mathbb{S}$ be an interaction model over $(\mathcal{P}, \mathcal{M}sg)$ and $\mathbb{A}$ be an activity model over $\mathcal{P}$. $\mathbb{A}$ *conforms to* $\mathbb{S}$, denoted $\mathbb{A}$ *tioco* $\mathbb{S}$, if and only if for every $\sigma \in TTraces(\mathbb{S})$ and $r \in O(\mathbb{S}) \cup D$,

$$\forall \sigma' \in TTraces(\mathbb{A}) : \sigma' \sim_{\mathcal{M}sg} \sigma.r \Longrightarrow \sigma.r \in TTraces(\mathbb{S})$$

Example. Let us consider the requirement **(R1)**. Applying the previous definition, the following trace of the activity model violates this requirement:
$(6ms).ctrl.intensity\_env?HIGH.(2ms).ctrl.intensity!HIGH.(1ms)$
$.\overline{calc.intensity}!HIGH.(3ms).calc.speed!FAST.(1ms).\overline{ctrl.speed\_calc}!FAST$
$.(6ms).ctrl.speed\_motor!FAST.(2ms).\overline{motor.speed}!FAST.(3ms)$
$.ctrl.intensity\_env?DRIZZLE.(484ms).ctrl.intensity!DRIZZLE.(1ms)$
$.\overline{calc.intensity}!DRIZZLE.(4ms).calc.speed!FAST$

The violation is due to the inappropriate calculated wiper speed. In fact, for the first detected amount of rain, high intensity, the fast speed is correct. At drizzle, the calculator computed again a fast speed, however the speed must be low.

5.2 Testing Process

We use the so-called *off-line testing* presented in [3] to test the conformance of the activity model, in the sense of *tioco*. The process starts with choosing a path in the symbolic tree as a *test purpose* which covers a specific requirement. Then using constraint solving techniques, the idea is to derive a sequence of concrete inputs and durations which would allow the execution of the activity model to potentially cover the test purpose. In order to submit the input sequence, a tester is connected to the system. The behavior of the tester is specified with an fUML activity in a textual form, ALF (Action Language for Foundational UML)[2]. This allows for automatic generation of tester behavior from input sequences. Repeatedly, the tester sends an input value on the targeted port of the system and then waits for the subsequent duration. See Figure 9 for illustration on the RSW system. Note that classically *tioco* assumes that *the system under test is input enabled*, i.e. that an SUT cannot refuse an input from the tester. This hypothesis is satisfied by running the tester behavior in the fUML virtual machine which is also used as the test harness in our framework. The other hypothesis of *tioco* is called *time elapsing* and expresses that the absence of an output amounts to observing no reaction from the system during a delay. As a discrete time simulator, the fUML virtual machine grants this hypothesis for consistent executions. Indeed after the initialization at the beginning of the execution, the simulated time elapses to reach the earliest waiting time specified either in the tester or in any component of the system.

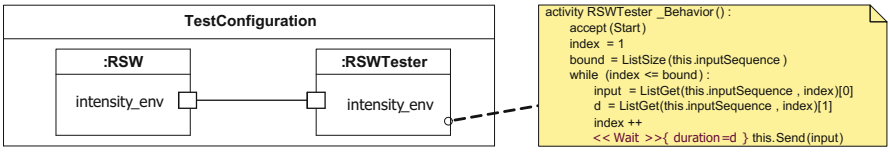


Fig. 9 RSW test configuration – tester activity in ALF syntax

As required by off-line testing, outputs and durations between them are logged during the test execution. This output sequence is merged with the input sequence to form a timed trace. In the final phase, the trace is analyzed w.r.t. the interaction model to emit a verdict: PASS, if we observe exactly the desired behavior; INCONC, if we observe a behavior that is not compatible

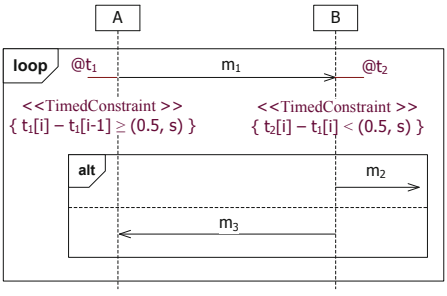
[2] `http://www.omg.org/spec/ALF/`

with the test purpose; and FAIL, if we observe an output or a delay that is not specified.

6 Experiments

Coverage and path explosion problem. We consider *message coverage* which is one of the criteria defined in the literature for scenario models [2]. It states that any message must be covered at least once. In order to achieve coverage, Diversity allows to define exploration strategies. Classical search algorithms like Breadth First Search (BFS) are implemented. However, using BFS results in exploring a large number of paths in the symbolic tree which are irrelevant to the coverage criteria. We suggest using the heuristic search called *Hit-or-Jump* [5] which computes a symbolic tree covering a declared set of transitions. In our case, its is a set of transitions matching emissions/receptions of the messages in the sequence diagram. first we define a maximal depth N for which a symbolic tree is computed in BFS manner. Once computed, an analysis is realized to study whether or not a part of the tree satisfies the coverage: (*Hit*) If some non empty prefixes of the sequence has been covered, Diversity identifies the set of paths that covered the greatest prefix, and chooses one among them at random else Diversity chooses at random one path; (*Jump*) Once a path is chosen the whole process starts again from the last symbolic state of the path (i.e. the target state of the last symbolic transition of the path) until the sequence is fully covered. Another version of the Hit-or-Jump (and more accurate as in [5]) tries to cover a set of transitions rather than an enforced sequence which is useful in some cases when it is not easy to predict an appropriate sequence of messages as illustrated in figure 10.

Note that introducing timing constraints may constrain the FIFO size. Recall that we associate to each message in the sequence diagram an unbounded FIFO buffer. In the sequence diagram of figure 10, the FIFO is similar to one-place buffer due to the timing constraints. Hence the sequence **(SEQ2)** can hardly be covered within a reasonable number of jumps regarding the size of



(SET1)
$\{A!m_1, B?m_1, B!m_2, B!m_3, A?m_3\}$
(SEQ1) $A!m_1.B?m_1.B!m_3$
$.A?m_3.A!m_1.B?m_1.B!m_2$
(SEQ2) $A!m_1.A!m_1.B?m_1.B!m_3$
$.A?m_3.B?m_1.B!m_2$

Input ($N = 2$)	Hit-or-Jump	#Jumps
(SET1)	covered	$5N$
(SEQ1)	covered	$> 5N$
(SEQ2)	timeout	$>> 5N$

Fig. 10 Hit-or-Jump/sample sequence diagram

the diagram (if the actions in the sequence must be consecutive, the Hit-or-Jump deadlocks). Let us consider again the sequence diagram of the RSW system. In Table 1, are given some metrics about the symbolic execution of its set of communicating TIOSTS using the heuristic Hit-or-Jump. *Failing behaviour.* While first testing the fUML activities of the RSW system w.r.t Requirement **(R2)**, Diversity delivered FAIL verdicts for some input sequences. The failure was due to intensity measures being ignored by the controller. Recall the fUML execution semantics in Section 4. The activity of the controller while waiting for the speed from the calculator was: checking the event pool, reading intensity measures and ignoring them. To solve this problem which is caused by the event handling in the fUML virtual machine, a separate buffering mechanism for measures was successfully introduced in the activity model.

Table 1 Hit-or-Jump/symbolic execution of the RSW system

#TIOSTS	#States	#Transitions
7	81	192

Input	N	time	#Jumps
(SET2)	7	$5s$	$2N$
(SEQ3)	7	$22s$	$7N$
(SEQ4)	3	$9s$	$14N$

Requirement **(R3)/(SET2)**
$\{ctrl.intensity!m_2, \ldots, motor.speed?m_4\}$
Requirement **(R2)/(SEQ3)**
$ctrl.intensity?m_1 \ldots ctrl.intensity?m_1$
$\ldots ctrl.intensity?m_1$
Requirement **(R1)/(SEQ4)**
$calc.intensity?m_2.new(calc.speed).calc.speed!m_3$

7 Related Works

We can find in recent literature approaches [14, 16] which have addressed conformance testing based on sequence diagram in the frame of the *ioco* relation (the untimed version of *tioco*). Authors in [16] use sequence diagrams in testing activities. They derive test cases expressed as sequence diagrams from state-based UML models guided by test objectives, also expressed as sequence diagrams. Authors in [14] have defined operational semantics for sequence diagrams where they handle in addition assertion and negative operators (*neg* and *assert*) for forbidden and required behaviors. However, they do not consider timing features in the test derivation algorithm. Let us discuss the approaches which deal with symbolic test generation from scenarios. Testing based on symbolic denotation of scenarios has been considered in [18] where scenarios are graphical MSCs (Message sequence chart) [12] like sequence diagrams. The test cases are experimented against the implementation within the frame of *ioco*. Outside *ioco* frame, symbolic techniques are used in [7] to generate test inputs from information contained in class diagrams and sequence diagrams. Transformation rules are defined to obtain a directed graph VGA (Variable Assignment Graph). The authors define coverage criteria for

sequence diagrams in order to select relevant paths and use solvers to compute test inputs. This work is closely related to ours since they use likewise generated inputs to test an executable form of the design models, however they do not consider timing constraints. Our approach is compliant with the lately standardized fUML virtual machine to execute activity models and an ongoing standardization of the semantics of UML composite structures [10]. Formal verification of fUML executable models has been studied in [1, 17]. We rather focus on testing fUML models as in [6, 15]. In particular, authors in [15] set up a test environment with an interpreter to run test cases in the fUML virtual machine. Our work is more complete because we integrate test generation capabilities from sequence diagrams.

8 Conclusion

In this paper, we have presented an approach which aims at enhancing confidence in the correctness of wide system models through refinement. The refinement is based on maintaining a correctness relation which states sufficient conditions on conforming executable models with respect to high-level timed scenarios. Our approach is tooled and compliant with UML standards. In the future, we plan to integrate more refinement techniques as in [8] and extend them to timing issues.

References

1. Abdelhalim, I., Schneider, S., Treharne, H.: Towards a practical approach to check UML/fUML models consistency using CSP. In: Qin, S., Qiu, Z. (eds.) ICFEM 2011. LNCS, vol. 6991, pp. 33–48. Springer, Heidelberg (2011)
2. Andrews, A.A., France, R.B., Ghosh, S., Craig, G.: Test adequacy criteria for uml design models. Softw. Test., Verif. Reliab. (2003)
3. Bannour, B., Escobedo, J.P., Gaston, C., Le Gall, P.: Off-line test case generation for timed symbolic model-based conformance testing. In: Nielsen, B., Weise, C. (eds.) ICTSS 2012. LNCS, vol. 7641, pp. 119–135. Springer, Heidelberg (2012)
4. Bannour, B., Gaston, C., Lapitre, A., Escobedo, J.P.: Incremental symbolic conformance testing from UML MARTE sequence diagrams: railway use case. In: HASE. IEEE (2012)
5. Cavalli, A., Lee, D., Rinderknecht, C., Zaïdi, F.: Hit-or-jump: An algorithm for embedded testing with applications to IN services. In: Wu, J., Chanson, S.T., Gao, Q. (eds.) FORTE. IFIP AICT, vol. 28, pp. 41–56. Springer, Heidelberg (1999)
6. Craciun, F., Motogna, S., Lazar, I.: Towards better testing of fUML models. In: ICST (2013)
7. Dinh-Trong, T.T., Ghosh, S., France, R.B.: A systematic approach to generate inputs to test UML design models. In: ISSRE. IEEE (2006)

8. Faivre, A., Gaston, C., Le Gall, P., Touil, A.: Test purpose concretization through symbolic action refinement. In: Suzuki, K., Higashino, T., Ulrich, A., Hasegawa, T. (eds.) TestCom/FATES 2008. LNCS, vol. 5047, pp. 184–199. Springer, Heidelberg (2008)
9. Object Management Group. A UML profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, VSL (2009), http://www.omg.org/spec/MARTE/
10. Object Management Group. Pscs: Precise semantics of uml composite structures, Second revised submission (2013) (to appear)
11. Object Management Group. Semantics of a foundational subset for executable uml models, fUML (2013), http://www.omg.org/spec/FUML/
12. ITU-TS Recommendation Z.120: Message Sequence Chart (MSC). Geneva (1997)
13. King, J.C.: A new approach to program testing. In: Proc. of Int. Conf. on Reliable Software (1975)
14. Lund, M.S., Stølen, K.: Deriving tests from uml 2.0 sequence diagrams with neg and assert. In: AST (2006)
15. Mijatov, S., Langer, P., Mayerhofer, T., Kappel, G.: A framework for testing UML activities based on fUML. In: MoDeVVaMoDELS (2013)
16. Pickin, S., Jard, C., Jéron, T., Jézéquel, J.-M., Traon, Y.L.: Test synthesis from UML models of distributed software. IEEE Trans. Software Eng. (2007)
17. Planas, E., Cabot, J., Gómez, C.: Lightweight verification of executable models. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 467–475. Springer, Heidelberg (2011)
18. Roychoudhury, A., Goel, A., Sengupta, B.: Symbolic message sequence charts. ACM Trans. Softw. Eng. Methodol. (2012)
19. Schmaltz, J., Tretmans, J.: On Conformance Testing for Timed Systems. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 250–264. Springer, Heidelberg (2008)
20. Tatibouet, J., Cuccuru, A., Gerard, S., Terrier, F.: Principles for the realization of an open simulation framework based on fuml (WIP). In: DEVS. ACM (2013)

MoSaRT Framework: A Collaborative Tool for Modeling and Analyzing Embedded Real-Time Systems

Yassine Ouhammou, Emmanuel Grolleau, Michaël Richard, Pascal Richard, and Frédéric Madiot

Abstract. The increasing evolution of real-time and embedded systems needs methodologies and design tools in order to reduce design complexity. Moreover, the scheduling analysis is one of the aspects that integrate the development process to reduce development costs and to validate systems. Since model-driven engineering offers interesting solutions to the abovementioned challenges, it is widely used in various industrial and academic research projects. This paper presents an overview of a model-based framework called MoSaRT (*Modeling oriented Scheduling analysis of Real-Time systems*), which aims to help real-time designers to conceive, dimension and analyze real-time systems. The underlying idea behind this proposal is to fill the gap between the academic real-time scheduling theory community and industrial practices. In fact, research results have been exploited in industrial contexts only to a modest extent to date. The MoSaRT framework is also a software tool for technology transfer enabling researchers to promote their works (e.g. analysis models and scheduling tests), then to increase the applicability of the real-time scheduling analysis.

1 Introduction

Real-time and embedded systems have been widely used in different industrial areas, like transportation, nuclear plants, and telecommunications. A

Yassine Ouhammou · Emmanuel Grolleau · Michaël Richard · Pascal Richard
LIAS lab. (ISAE-ENSMA and University of Poitiers) - Futuroscope, France

Frédéric Madiot
Obeo, France
e-mail: {ouhammoy,grolleau,richardm}@ensma.fr,
pascal.richard@univ-poitiers.fr, frederic.madiot@obeo.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_20

real-time system is a system that must interact with a correct behavior to input events within specified timing bounds [2]. So, a result that is functionally correct, but not temporally correct (i.e. not respecting the deadline), is considered as a wrong behavior.

We are interested in the temporal correctness of hard real-time systems. A hard real-time system has to meet its timing requirements (i.e. in order to be schedulable), otherwise, something unacceptable and catastrophic can occur. The hard real-time system is composed on a set of tasks and messages sharing a set of execution/communication resources. The way of sharing resources depends on the scheduling algorithms, the network protocols and the memory access policies which are chosen by designers. To check if the used resources/policies/protocols are enough and well adapted for that tasks and messages always meet the timing requirements, the "scheduling analysis" is applied during the design phase not only to check the schedulability of hard real-time systems, but also to help designers to dimension system's architecture when the system design is not completely defined. The real-time scheduling analysis can be based on the model checking, the simulation or the analytical methods of the scheduling theory.

Nowadays, the utilization of the real-time scheduling theory in practical cases could be profitable. Unfortunately, it is not sufficiently applied and the research results have been exploited in the industry only to a modest extent to date. The chasm between the scheduling theory and the industrial practices may be due to several reasons. For instance, to master the scheduling techniques, a colossal work related to the real-time theory knowledge is required. However, systems designers may not be well versed in the real-time scheduling theory. Industrial designers are too busy to perform accurate analyses due to cultural and economical reasons (e.g. concurrency/competitiveness), and they are unwilling to take risks with new approaches. Furthermore, transferring the knowledge from the research area to an industrial area can be expensive. Even if many analysis tools exist, a tool cannot offer all the analysis models and techniques. Moreover, whereas an academic researcher develops a prototype easing the exploitation of a special research study, adding this prototype in different analysis environments may require to modify their internal structures. That needs a high development effort for a research group other than the original tool makers.

Our objective is to help designers to cope with the analysis difficulty, then to orient them in order to choose the most appropriate analysis tests and to ease the design modifications due to refinement or dimensioning actions. Therefore, designers will be guided to build scheduling-aware models. The second objective is to enhance the applicability of the real-time scheduling theory. Therefore, it is needful to provide an easy way for transferring the research studies from academia to industrial practices. To achieve the above objectives, we propose a framework called MoSaRT (Modeling-oriented Scheduling analysis of Real- Time systems). MoSaRT is also an intermediate framework between real-time design languages and schedulability analysis tools. In this paper, we

gather different parts of the MoSaRT framework which have been already published separately [14] [15] [13], then we present to readers a global view of this framework.

The rest of the paper is organized as follows. In the next section, we give a brief overview of real-time scheduling concerns. Section 3 gives a general idea introducing the MoSaRT framework. Section 4 describes the MoSaRT design language. Section 5 presents the MoSaRT analysis repository. Section 6 highlights some typical MoSaRT usage scenarios. Finally, Section 7 summarizes and concludes the paper.

2 Background and Related Work

Since 1970s, researchers of the real-time community have presented several research works dedicated to the scheduling analysis of hard real-time systems [18] [3]. On the one hand, these works consist on a set of analysis models. Indeed, an analysis model represents formal expressions admitting mathematical evaluations and based on temporal properties taking into consideration different task characteristics (like the precedence relationship or the self-suspension) and hardware architectures (uniprocessor, multi-cores processors, distributed systems, etc.). On the other hand, the research works have also tackled various analysis tests helping designers (especially analysts) to check the temporal validation of the real-time applications. While every analysis model is an extraction of the non-functional temporal properties from a system design, then the analysis tests depend on the analysis models. In fact, the kind of the analysis tests depends on the completion stage of the system design (i.e. does the system design need to be dimensioned or validated ?). Moreover, the efficiency and the consistency of an analysis test depend on the design and temporal characteristics of the system.

The steep learning curve behind many of the current analysis methods has been one of the major impediments to their adoption and their exploitation in the industry. Several works have treated the difficulty of the scheduling analyses utilization through a model-based engineering process. They proposed modeling languages and tools to decrease this difficulty. Recently, UML-MARTE [12] and AADL [1, 10] are among standard modeling languages that have been proposed. MARTE (Modeling and Analysis of Real-Time and Embedded Systems) is a UML profile that offers several stereotypes and tagged values helping designers to annotate their UML models (e.g. class diagrams). The purpose is to add temporal characteristics and constraints for further scheduling analyses. However, since UML-MARTE does not follow a specific standard methodology, semantics of the stereotypes differ from an utilization to another. Hence, a set of methodologies have been proposed using only a subset of UML-MARTE and with different semantics (like Optimum [9] and MADES [17]). The underlying idea behind those methodologies is not only to ease the utilization of a subset of UML-MARTE, but also to help designers

by proposing a task-set (i.e. design patterns of the tasks architecture) that fits with the functional model. AADL (Architecture and Analysis Description Language) is a component-based language leading to get hierarchical system architectures close to the reality, containing a set of hardware and software components. Although AADL does not allow designers to define the functional part of real-time systems, the architectures are expressed in a modular way. Nevertheless, the utilization of AADL only through the development life-cycle of embedded system can not help to get refined models iteratively. In other words, AADL does not enable to dimension models (e.g. allocation of tasks, mapping of functions, partitioning).

The implementation of the analysis techniques has also taken advantage of the model-driven engineering. Recently, several academic and industrial tools were proposed as providers of the well-known analysis techniques by offering the possibility to apply some subsets of schedulability tests during the design phase. Some examples of those tools are RT-Druid [4], SymTA/S [5] and Cheddar [19]. The utilization of the analysis tools provides often a simple Yes/No answer to the question "does the system meet all its deadlines?". This kind of information is not efficient and not helpful enough for real-time designers, in particular, when the analysis tool is not able to analyze the system. Moreover, analysis tools do not help designers to choose the appropriate tests which match the model requesting the analysis. So, even if the analysis result is provided, it may be very pessimistic due to the choice of a wrong analysis test.

3 Objectives of MoSaRT Framework

As the modeling and the scheduling analysis of real-time systems are both in constant evolution and improvement in distinct scientific communities, the design methodologies, design languages and analysis tests are sharply improved. Indeed, the methodologies are impacted by the hardware equipments, the software operating systems and the programing languages. While the model-driven engineering offers a relative independence regarding technological changes, and provides a re-usability of the design elements, that are measurable, predictable, and manageable, hence we are based on the model-driven engineering: (i) to unify modeling and analysis efforts,(ii) to achieve a friendly utilization taking benefits from standard design languages and timing analysis tools and (iii) to increase the applicability of the real-time scheduling theory. Consequently, we propose an intermediate framework named MoSaRT (Modeling-oriented Scheduling analysis of Real-Time systems). To partition the efforts, the intermediate framework plays the role of a bridge between the real-time design languages and the analysis tools (see Figure 1). It is based on two metamodels interacting with each other:

– MoSaRT Design Language, which is a domain specific language offering enough concepts and semantics to obtain models independent from any methodology, and to cover with few modifications, most existing analysis models and easily extended to include concepts enabling to support future notions.
– MoSaRT Analysis Repository metamodel allows analysts to plug different theoretical studies and prototypes and ensures the interoperability with different analysis tools in order to compare their output results.

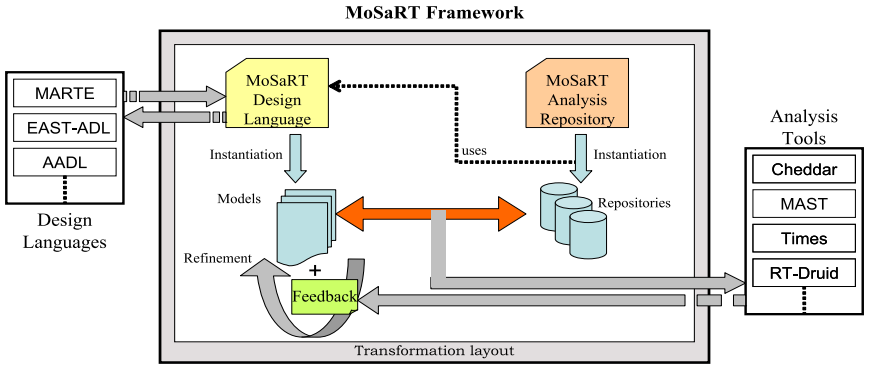


Fig. 1 MoSaRT Framework

Figure 1 gives only an overview of our contributions. It shows a generic scenario due to the usage of the MoSaRT framework. This latter helps designers to analyze step by step their system designs during the design phase, which can be increasingly improved by applying the three following processes:

– Using the MoSaRT design language for system modeling, or for refining imported models.
– Selection of the analysis models and the relevant tests, by helping the real-time designers to extract the relevant elements from the models.
– Scheduling analysis, by offering to the real-time designers the equipped analysis tests which correspond to their models via the proposition of one or several analysis repositories. The next sections discuss the details of each contribution.

4 MoSaRT Design Language

MoSaRT design language [14] [15] is conceived as a domain specific modeling language for real-time systems. It contains several concepts which are very close to the real-time analysis. The MoSaRT language is based on the notion of viewpoints complementarity by proposing different kind of models: hardware model, software architecture model, behavioral model and functional

model. The implementation of MoSaRT language is based on Ecore language [20] and Sirius (see Section Acknowledgment).

The MoSaRT language generic real-time properties: every real-time concept (like execution-time property) has been meta-modeled to support different kinds of systems in different design stages. Furthermore, every model expressed in MoSaRT language can be checked by several structural rules implemented in OCL (Object Constraint Language) [11] ensuring the vivacity, the safety and architectural correctness.

Figure 2 shows a hardware architecture of a real-time system. The system is composed of two nodes communicating through a CAN (Controller Area Network) network. The first node is uniprocessor, and the second node is a multi-core processor containing four cores. The software architecture

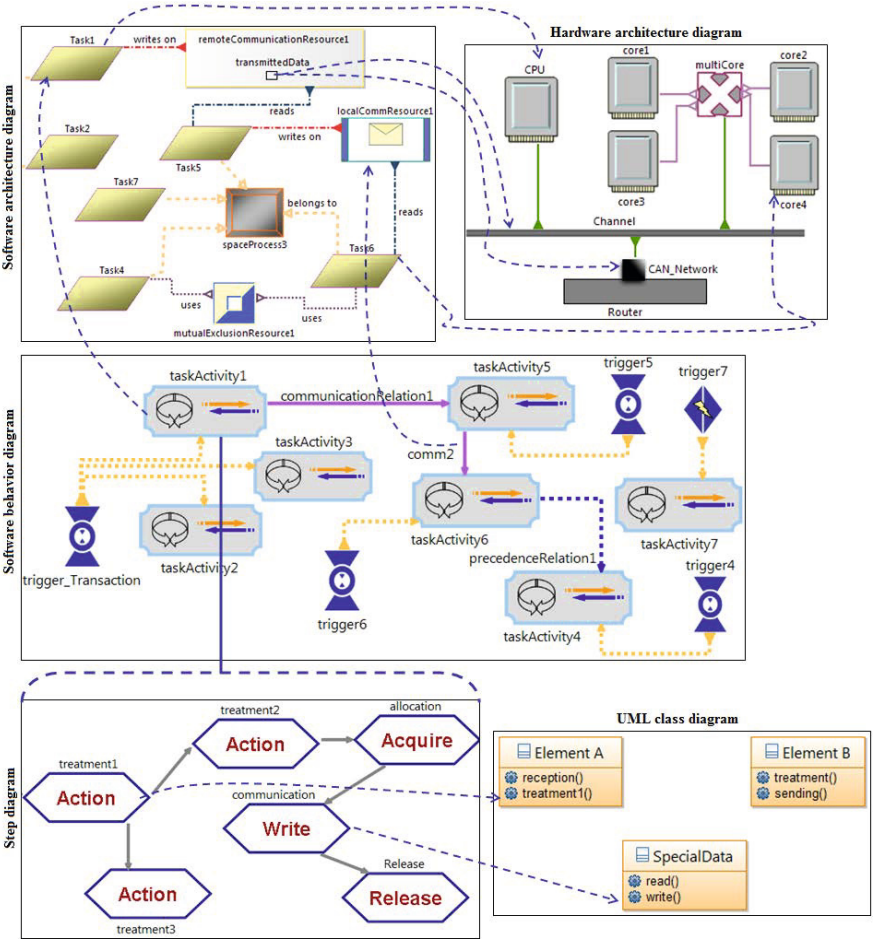


Fig. 2 Different steps of the identification process

diagram of Figure 2 represents a software architecture model that contains seven tasks managed by three schedulers, and two of them are hierarchical. It also contains two interaction resources shared by three tasks (mutual exclusion resource and box communication resource), and a remote communication resource allowing to transmit data between two tasks. The software architecture model can be mapped to different behavioral models. The one shown in Figure 2 represents the global behavior of the system and requires a root trigger meaning the timing reference of the remainder triggers. The model contains several task activities. A task activity can be triggered by its own trigger or it can be triggered by another task activity. The precedence relationship represents a synchronization between the task activities. The existence of a communication relationship between two task activities implies the existence of a shared communication resource in the software architecture model. Through the behavioral model, the content of every task activity can be defined thanks to the step diagram. Every step describes the elementary actions of the task activity like the read action, the release action, etc. The importance of step elements is also their capability to allocate the operational side of a real time-system (hardware, software architecture and behavioral models) to the functional side (e.g. UML models).

5 MoSaRT Analysis Repository

We have noticed the absence of an instrumented method guiding the designers to the best model and tests for their systems. Moreover, the passage from the system modeling to the system analysis requires dual skills, in order (i) to identify the appropriate analysis situation of the system design and (ii) to find the suitable analysis tests. We note, $Ar = <\mathcal{R}, \mathcal{X}, \mathcal{G}, \mathcal{T}, E>$ is the MoSaRT analysis repository, where:

- $\mathcal{X}$ is a set of real-time contexts. Every real-time context represents a set of specific assumptions, where each assumption is related to the software architecture, the timing behavior or the hardware architecture. The real-time context represents the analysis situation to which the system design corresponds. In other words, thanks to the real-time context we can know the analysis model that matches the system design.
- $\mathcal{G}$ is a set of generalization relationships between some real-time contexts. The generalization relationship is an order-relation treated in the real-time scheduling theory. A real-time context "a" is a generalization of the real-time context "b", if the behavior of "a" includes all possible behaviors of "b".
- $\mathcal{T}$ is a set of analysis tests. Every analysis test is based on a real-time context. When it is applied to a system, the result provided by the test is correct if the system respects all the context assumptions.
- E is a set of analysis tools. The analysis tool is an engine proposing an independent functionality inside an analysis framework (like MAST or

Cheddar). The same analysis functionality inside another analysis framework is considered as another engine.
– It is common to find several real-time context characterized by the same subset of assumptions, or the opposite subset of assumptions. So, for factoring the number of assumptions and to guarantee a good scalability of the analysis repository, we suggest that the analysis repository contains also a set of identification rules $\mathcal{R}$. They will help for identifying correctly the closest real-time context matching the design model which requests analysis. Every identification rule is characterized by a formal expression as an OCL constraint, this latter depends on MoSaRT design language.

5.1 Instantiation of the MoSaRT Analysis Repository

The metamodel of the MoSaRT analysis repository (see Figure 3) is dedicated to be instantiated by schedulability-aware theorists/analysts in order to obtain analysis decision supports. Thus, the designer's orientation will be based on the richness and the correctness of the analysis decision support. This latter may be fed and enriched by theorists/analysts (i) to compare their results with existing ones (already stored in the decision support), (ii) and to facilitate the use of their works (and their prototypes) by designers.

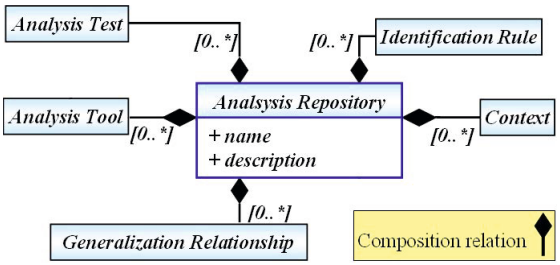


Fig. 3 Analysis repository Metamodel

In order to create a new repository instance from scratch, we start by instantiating the `IdentificationRule` to get a set of identification rules. Next, we instantiate the `Context` to create a real-time context based on the existing identification rules. Furthermore, we add to the repository instance some tests and analysis tools corresponding to the created real-time context.

First of all, we provide an analysis repository containing a real-time context corresponding to the Liu and Layland model [8]. This context is based on several identification rules (some of them are shown in Figure 4). Each rule is mapped to a formal expression implemented as an OCL constraint related to the design language of the system that needs analysis (i.e. the MoSaRT design language). Figure 4) shows the formal expression of the identification

rule called "`UniprocessorArchitecture`". Moreover, we have chosen the response time analysis test presented in [6] as an analysis test for the context corresponding to the analysis model of [8]. This test is implemented by several tools like Rt-Druid [4].

Furthermore, we have added the real-time context of the transaction model [16]. This latter represents a "generalization" of the periodic model which had been previously created in the analysis repository. A second response time analysis test is added to the repository. It is devoted to analyze the transaction model. We mention MAST as a provider of this second test. Besides, we have embodied the generalization relationship, and connect it with a transformation program enabling the transition from the periodic model to the transaction model (when it is possible). The transformation program is done with ATL (Atlas Transformation Language) [7]. It represents an endogenous transformation (i.e. a transformation from MoSaRT design language to MoSaRT design language).

6 MoSaRT Usage Scenarios

6.1 The Back-end of the MoSaRT Framework

Figure 5 indicates the back-end and the front-end of the MoSaRT framework. MoSaRT back-end provides to analysts various capabilities. So, one may propose a new analysis repository (related to a specific company/laboratory) by instantiating the analysis repository model (Action (1) of Figure 5). In

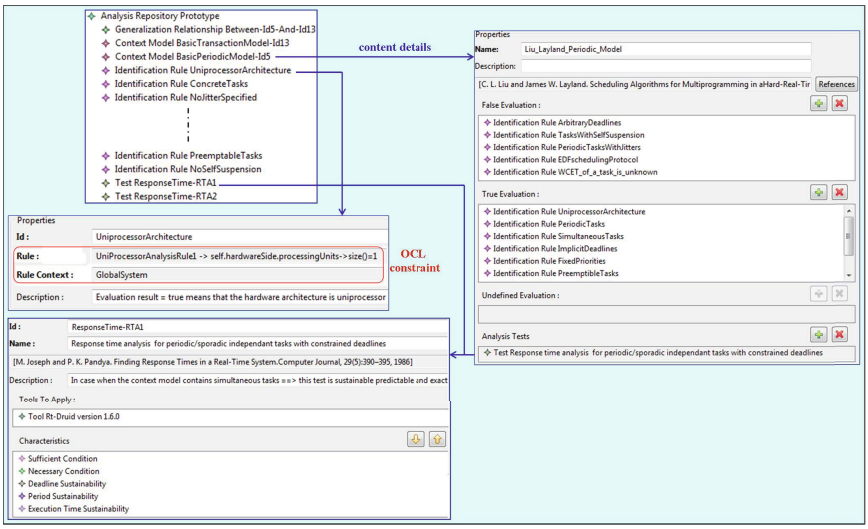


Fig. 4 Part of an analysis repository highlighting the details of a context

this case, one should define at least the contexts and tests to guarantee a minimum usability of the repository. To rise the usability and to get a full coherence of an analysis repository, this latter can be enriched progressively by adding new contexts, new tests, new tools, etc (Action (2) of Figure 5). Therefore, every real-time context already existing in such a repository can be refined by specifying more accurate identification rules, more characteristics of the analysis tests, and automatizing the transformation to analysis tools, etc.(Action (3) of Figure 3). Once an analysis repository becomes ready for use, it can be shared in order to be used by designers (Action (4) of Figure 5).

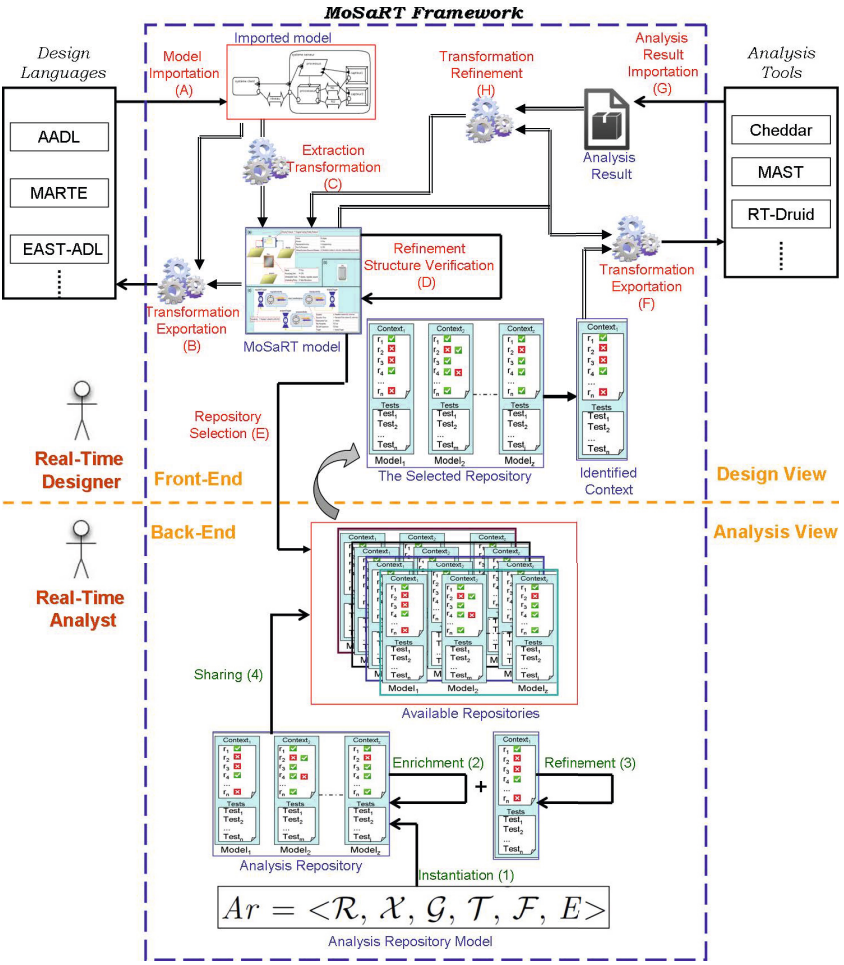


Fig. 5 Some of the relevant utilization scenarios related to the MoSaRT Framework

6.2 The Front-end of the MoSaRT Framework

The MoSaRT front-end is totally related to the MoSaRT design language. Figure 5 gives an overview of some capabilities. The design models expressed in a standard language like MARTE can be imported (Action A of Figure 5). Then, designers can use the transformation process offered by MoSaRT framework in order to transform their models to MoSaRT design language. The transformation process from a standard language to MoSaRT language is based on the extraction of timing details (Action C of Figure 5). While the MoSaRT framework gives the possibility to use its modeling environment, actions A and C are not mandatory. Once designers obtain models expressed in MoSaRT language, they can refine them and check the correctness of their structure (Action D of Figure 5). Hence, the analysis stage starts when designers select an available MoSaRT analysis repository (Action E of Figure 5). When identifying the corresponding real-time context (if the repository is rich enough), a customizable transformation to analysis tools can be provided (Action F of Figure 5). Once getting the analysis result, MoSaRT framework gives the possibility to import the tool output files (Action G of Figure 5). Due to technical transformation reasons, the Action H of Figure 5 requires both the original MoSaRT model and the analysis result file. The refinement and the analysis actions can be repeated many times until getting accurate models. The transformation of the MoSaRT analyzed model to an external design language can be done only if the model was imported (Action B of Figure 5).

7 Conclusion

By proposing the MoSaRT framework, our objective is to benefit from the analyst's skills and designer's skills in order to unify their efforts, then to avoid wrong design choices at an early design phase. The framework is based on the MoSaRT design language and the MoSaRT analysis repository, and presents a helpful modeling support for both designers and analysts. Since we have tried to facilitate the use of concepts based on theoretical studies and dedicated to a concrete industrial utilization, we may consider the MoSaRT framework as a tool for technology transfer.

We are working to enrich the MoSaRT Framework, in particular, in case of the non schedulability of the system, currently the MoSaRT framework proposes only a dimensioning analysis if it exists (it depends on the system context). However, some works like Optimum (applied to UML-MARTE) proposes to change down-right the system architecture if the functional model exists. Such works can be connected to the framework in order to be called after the restitution step (e.g. obtaining the analysis result). It will be also helpful to generate a design expressed in MoSaRT language by choosing a priori the real-time context to which the design corresponds. In this case,

the real-time context is used as a design pattern. For example, generating a design model respecting the pattern "Ravenscar Profile" which is widely used in industry can be very interesting.

Acknowledgment. We thank Obeo[1] that has provided us the Sirius product which is a new Eclipse project (http://www.eclipse.org/sirius). We used this tool to implement the concrete graphical syntax of the MoSaRT design language. It is particularly adapted to define a DSL (Domain Specific Language) that needs graphical representations to better elaborate and analyze a system and improve the communication with other team members, partners or custom. All shape characteristics and behaviors can be easily configured with a minimum technical knowledge. This description is dynamically interpreted to materialize the workbench within the Eclipse IDE. No code generation is involved, the specifier of the workbench can have instant feedback while adapting the description. Once completed, the modeling workbench can be deployed as a standard Eclipse plugin. Thanks to this short feedback loop a workbench or its specialization can be created in a matter of hours.

References

1. W. SAE AADL. The SAE Architecture Analysis & Design Language Standard, vol. 2009 (2009)
2. Burns, A., Wellings, A.: Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX, 4th edn. Addison Wesley (2009)
3. Davis, R.I., Burns, A.: A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43(4), 35 (2011)
4. Gai, P., Natale, M.D., Serreli, N., Palopoli, L., Ferrari, A.: Adding timing analysis to functional design to predict implementation errors. SAE Technical Paper 2007-01-1272 (2007)
5. Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., Ernst, R.: System level performance analysis–the symta/s approach. IEE Proceedings-Computers and Digital Techniques 152(2), 148–166 (2005)
6. Joseph, M., Pandya, P.K.: Finding response times in a real-time system. Computer Journal 29(5), 390–395 (1986)
7. Jouault, F., Kurtev, I.: Transforming models with ATL. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
8. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM 20(1), 46–61 (1973)
9. Mraidha, C., Tucci-Piergiovanni, S., Gerard, S.: Optimum: a marte-based methodology for schedulability analysis at early design stages. ACM SIGSOFT Software Engineering Notes 36, 1–8 (2011)
10. Society of Automotive Engineers (SAE). The SAE architecture analysis & design language standard, `http://www.aadl.info` (last access: April 15, 2014)
11. Object, O.: constraint language, omg available specification, version 2.0 (2006), `http://www.omg.org/spec/OCL/2.0/`
12. OMG. Uml profile for marte: Modeling and analysis of real-time embedded systems (2009), `http://www.omgmarte.org`

[1] `www.obeo.fr`

13. Ouhammou, Y., Grolleau, E., Hugues, J.: Mapping aadl models to a repository of multiple schedulability analysis techniques. In: IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), p. 8 (2013)
14. Ouhammou, Y., Grolleau, E., Richard, M., Richard, P.: Model driven timing analysis for real-time systems. In: IEEE International Conference on Embedded Software and Systems (ICESS), pp. 1458–1465 (2012)
15. Ouhammou, Y., Grolleau, E., Richard, M., Richard, P.: Reducing the gap between design and scheduling. In: Real-Time and Network Systems (RTNS), pp. 21–30. ACM (2012)
16. Palencia, J.C., González Harbour, M.: Schedulability analysis for tasks with static and dynamic offsets. In: IEEE Real-Time Systems Symposium (RTSS), pp. 26–37 (1998)
17. Quadri, I.R., Brosse, E., Gray, I., Matragkas, N.D., Indrusiak, L.S., Rossi, M., Bagnato, A., Sadovykh, A.: Mades fp7 eu project: Effective high level sysml/-marte methodology for real-time and embedded avionics systems. In: International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), pp. 1–8 (2012)
18. Sha, L., Abdelzaher, T., AArzén, K.-E., Cervin, A., Baker, T., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., Mok, A.K.: Real time scheduling theory: A historical perspective. Real-Time Systems 28(2-3), 101–155 (2004)
19. Singhoff, F., Plantec, A., Dissaux, P., Legrand, J.: Investigating the usability of real-time scheduling theory with the cheddar project. Real-Time Systems 43(3), 259–295 (2009)
20. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework. Pearson Education (2008)

Preliminary Hazard Analysis Generation Integrated with Operational Architecture – Application to Automobile

Pierre Mauborgne, Samuel Deniaud, Eric Levrat, Eric Bonjour, Jean-Pierre Micaëlli, and Dominique Loise

Abstract. We are witnessing evolution of standards (as the functional safety one) and increasing of complexity. This implies to perform safety studies efficiently and earlier in the context of Model-Based System Engineering. So, in this article, we will propose an evolution of the Preliminary Hazard Analysis (PHA) method in order to comply with the overall safety requirements in the automotive domain. To demonstrate its usefulness, we apply this method to an industrial case which concerns the hazard analysis of unintended acceleration of a vehicle.

Keywords: MBSE, Safety, Operational Architecture, PHA, ISO 26262.

Pierre Mauborgne · Dominique Loise
PSA Peugeot Citroën – route de Gisy, 78140 Vélizy-Villacoublay, France
e-mail: {pierre.mauborgne,dominique.loise}@mpsa.com

Pierre Mauborgne · Eric Bonjour
Université de Lorraine/ENSGSI, ERPI,
EA no 3767, 8, rue Bastien Lepage, Nancy, 54010, France
e-mail: {pierre.mauborgne,eric.bonjour}@univ-lorraine.fr

Samuel Deniaud
IRTES-M3M, UTBM, Université de Technologie de Belfort-Montbéliard, 90010 Belfort Cedex, France
e-mail: samuel.deniaud@utbm.fr

Eric Levrat
Université de Lorraine, Centre de Recherche en Automatique de Nancy (CRAN) - CNRS, UMR 7039, Campus sciences, BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
e-mail: eric.levrat@univ-lorraine.fr

Jean-Pierre Micaëlli
Université de Lyon, INSA Lyon, Centre des Humanités, 1, rue des Humanités, 69621 Villeurbanne Cedex, France
e-mail: jean-pierre.micaelli@insa-lyon.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_21

1 Introduction

Currently, automotive functional safety standards require manufacturers to demonstrate vehicle safety. Moreover, cars are increasingly complex due to the integration of innovative functions and the respect of regulations related to emission, safety, etc. Activities that aim at evaluating safety properties are often performed separately from design activities. Model-Based Systems Engineering is an opportunity to integrate safety analysis efficiently with Systems Engineering. The expected results are more efficient development. However Systems Engineering and Safety activities have specific methods and tools but also some similar concepts. In order to bridge the gap towards Safe Systems Engineering, a common terminology is required.

In this article, we will focus on the activity initiating the Safety analysis process, which is the Preliminary Hazard Analysis (PHA). It is possible to retrieve the content of this activity as "Hazard and Risk Analysis" in the functional safety standards (1), as "Aircraft (or System) PHA" in aeronautical standards (2) or as "Hazard Analysis and Risk Assessment" in the automotive standard (3).

For Desroches et al. (4), this activity is used to determine a classification of hazardous situations. This is very important because it influences further activities of the safety process.

This article proposes a conceptual model of safe systems engineering with concepts relevant for the PHA. After defining the process of the Concept of Operations and Requirements Analysis, we will explain a method related to the “Preliminary Hazard Analysis” process. Two kinds of PHA are identified: functional PHA, threat-related PHA. Finally we will apply this method to a specific case that concerns an automotive vehicle. As this method is applicable in other domains, we will indicate when the concepts are specific to the automotive domain.

2 Background on Links between Safety and Systems Engineering

Clear relations between systems engineering and safety is a prerequisite for an efficient integration of engineering and safety analysis.

We can distinguish two types of approaches that deal with these relations: model transformation (5), (6) and processes.

Yakimets et al. (5) selected several target languages as AltaRica (7) or NuSMV (8) which are Safety languages to perform safety analysis. After annotation of functional SysML diagrams, they realize a translation of SysML models to the target model.

Papadopoulos et al. (6) choose to focus on the generation of fault trees and FMEA from structure diagrams such as SysML (9) Internal Block Diagrams (IBD). The enrichment of blocks by “local” analysis and the links between these blocks enable to propagate failures and to get reports.

Regarding process approaches, Systems Engineering standards as ISO 15288 (10) indicate additional activities regarding safety, however these activities are realized in parallel of other activities of the process.

David et al.(11) has identified a method to perform FMEA by determining the necessary information either in the functional model or in a specific database (i.e. generic failure modes). By extending this approach, Cressent et al. (12) integrated it into a design process.

However, we can note that the standards evocate very few links. Moreover, we have found no work concerning a better integration of PHA and systems engineering. For this reason, we choose to focus our contribution on this key activity for the safety analysis.

Regarding "Preliminary Hazard Analysis" methods, the automotive functional safety standard (3) specifies inputs (item definition report), a sub-process and deliverables. It doesn't clarify the links between functional analysis and PHA. We may conclude that in ISO 26262 (3), this activity is subsequent to the functional analysis.

Desroches et al. (4) set out three steps to achieve a PHA: identifying dysfunctional scenarios, assessing them and proposing a risk coverage (or mitigation) that could be avoidance scenarios.

The objective of our contribution is to propose a process that integrates Systems Engineering and Safety activities but it does not concern transformations of model or language. We will present a process in detail which meets the objectives of PHA based on Systems Engineering models.

3 Conceptual Model of Safe Systems Engineering – Requirement Analysis View

3.1 Introduction

According to Rausand (13), there is no systematic process to realize a PHA. A cause of this statement is that different domains have their own concepts and different definitions for the same concept.

We also agree on the fact that some concepts have relative definitions and their perimeters are not necessarily well defined. It motivates the definition of the conceptual model of safe systems engineering. Table 1 defines 5 major requirements for this conceptual model.

Table 1 Requirements for the conceptual model of safe systems engineering

Reference	**Requirement.** The conceptual model shall
1	show the links between system engineering and safety.
2	be simple
3	enable to verify the correct application of a safety process.
4	provide a shared common language between the different stakeholders (experts, systems architects, safety engineers)
5	be compliant to the state-of-the-art methods of safety analysis

There are existing conceptual models that do not meet these requirements. The conceptual model in Chalé et al. (14) deals with links between systems engineering and ISO 26262 (3) but it is too generic to satisfy requirements 3 and 5. There is a unique view and concepts are not classified according to the processes. Moreover dependability concepts are in the same group of concepts and are not distribute in systems engineering groups.

The conceptual model in Deniaud et al. (15) contains some relevant concepts but we cannot rely a process on this model: Safety concepts are linked to systems engineering concepts. Conceptual model structure separates design and specification concepts. Due to that, it will be the basis for our conceptual model.

As Chalé et al. (14) or Deniaud et al. (15), the conceptual model we propose in this section is based on the analysis of key standards: IEEE 1220 (16), ISO 15288 (10) for Systems Engineering concepts and ISO 26262 (3) for safety concepts. In order to fulfill the above requirements, the model is divided into three views corresponding to the three main processes of system design (Requirements Analysis, Functional Architecture Design and Physical Architecture Design). In this article, we only present a partial view of the conceptual model of the first process, Concept of Operations and Requirements Analysis (Figure 1). It represents safety concepts and links to systems engineering concepts.

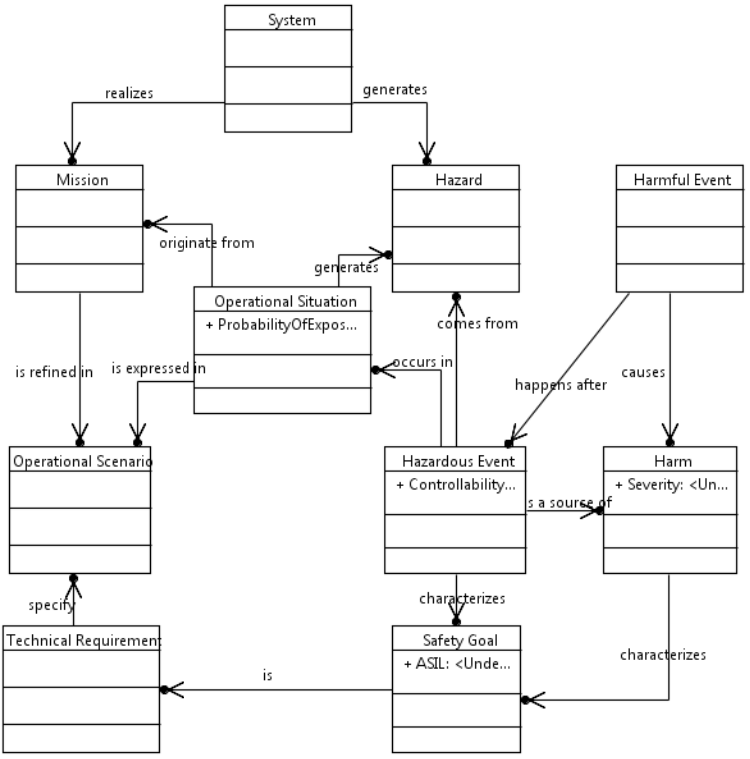


Fig. 1 Conceptual Model of Safe Systems Engineering –Requirement Analysis View

3.2 Safety Concepts Definition

If we cut the figure 1 into 4 columns, the two first columns are associated to the main concepts of systems engineering. We focus here on defining the concepts of the two last columns.

We can distinguish concepts to define the dysfunctional scenario and those for the assessment of the scenario.

Determination of Dysfunctional Scenario

Figure 2 presents the sequence of concepts about dysfunctional scenario.

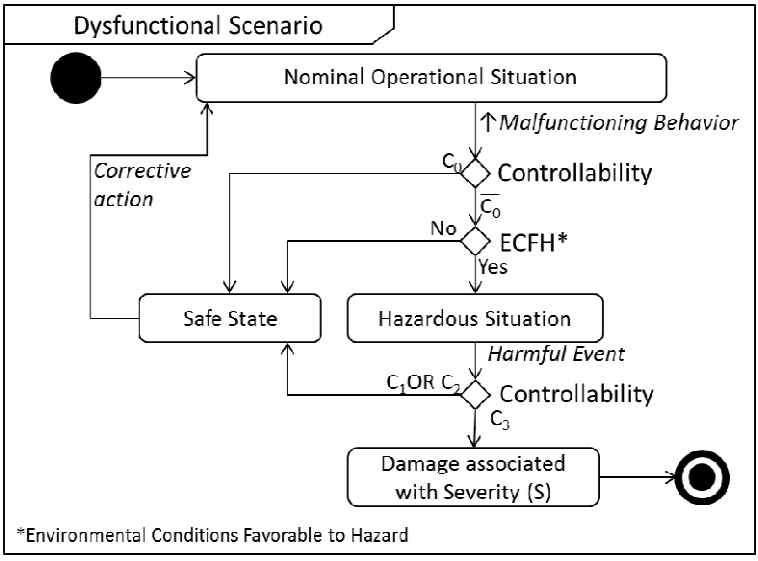


Fig. 2 Proposal for determining and assessing dysfunctional scenario

As defined in ISO 26262 (3), the hazard is a potential source of harm caused by a malfunctioning behavior of the system.

The apparition of these hazards (within environmental conditions favorable to hazard (ECFH) corresponds to a hazardous event, that generates a hazardous situation. A harmful event (such as a shock) triggers the transition from a hazardous situation to harm (damage associated with severity).

Dysfunctional Scenario Assessment

As stated before, we use ISO 26262, automotive functional safety standard (3) as a reference but most of the concepts are also present in other standards.

As Desroches et al. (4) defined, one step of the PHA concerns the assessment of dysfunctional scenario.

The probability of apparition of hazardous event is determined either qualitatively or quantitatively depending on the domain. If the qualitative criterion is used, the probability of apparition is defined by 4 classes in automotive domain.

Similarly, the severity of damage is evaluated for each situation, a level of severity can be associated.

Finally, in the automotive field, the controllability determines whether the driver can control the vehicle once he perceives the hazardous event. There are 4 levels to characterize this criterion (C0, C1, C2, C3).

The combination of these three criteria enables to define a Safety Integrity Level named ASIL in ISO 26262 (3). There are similar Safety Integrity Levels in other domains as SIL in IEC 61508 (1) or DAL in ARP 4761 (2) but there are different criterions to determine it.

From this analysis, we can specify a high level safety requirement. The safety integrity level is associated to the requirement. In automotive domain, these requirements are called Safety Goal in ISO 26262 (3) associated with ASIL.

4 Our Proposal Concerning the Integration of PHA and MBSE

The proposed conceptual model enables to have a common language between system architects and safety engineers. In this section, we present a process that integrates PHA and systems engineering models.

4.1 Inputs and Outputs of the PHA

PHA will be linked to the process of Concept of Operation and Requirements Analysis defined in ISO 15288 (10). There are similar processes in other references such as IEEE 1220 (16) or SEBoK (17).

The aim of PHA is to identify and classify hazardous events that can then be expressed as a safety goal. The objective of this activity is to provide the necessary inputs for the next activities of safety related to the functional and physical architectures.

These requirements are derived from the dysfunctional scenario, its assessment and possible avoidance scenarios. To define the dysfunctional scenario, events and situations must be defined (figure 2).

In the following sections, we will explain the global approach which is the most commonly used one. This approach is based on hazards which are deviations of output flows of the system to its environment.

After, we will adapt it to complete this analysis by taking into account the threats of the system towards the environment. Identification of hazards from this approach is based on the determination of emerging and abnormal flows.

4.2 Global Approach

Figure 3 shows an overview of the global approach.

First we have to find the possible beginnings and ends of the dysfunctional scenario.

With the diagram defining services of the system, we can have flow deviations for services output. Services explicit principal mission of the system. To complete the identification of hazards, the context diagram defining interfaces and flows between system and its environment is necessary. The deviation flow can be associated with the abnormal modification of a performance of a system service. This association allows having a clear and measurable safety requirement.

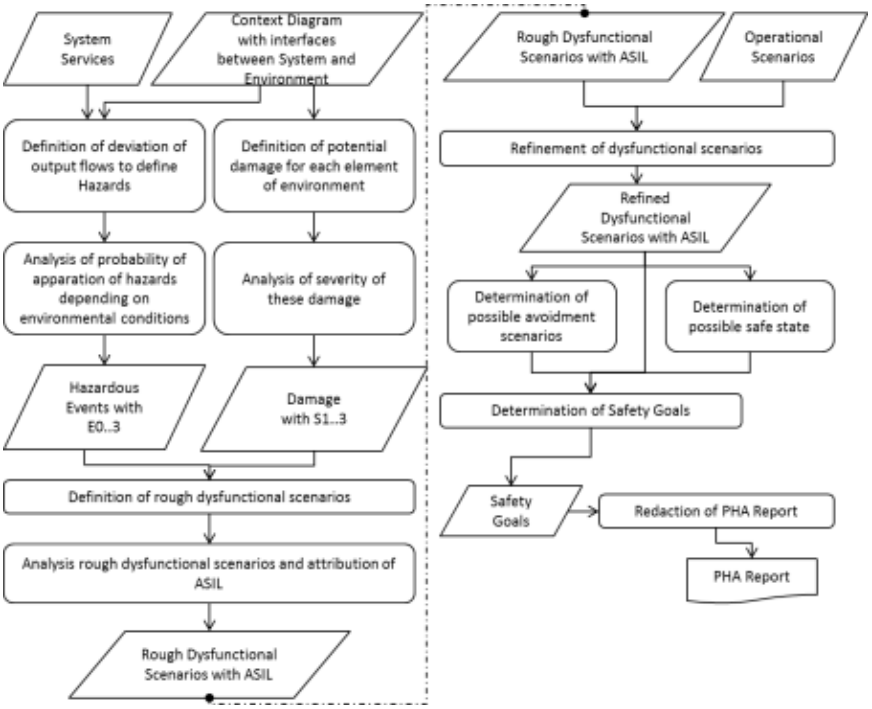


Fig. 3 Flowchart of the proposed approach

We identify the possible operational state of each element of the environment of the context diagram. For example, for a pedestrian, possible operational states are injured, safe or burnt. Possible damage is obtained by combinations of these states and events which trigger to damage.

We can evaluate them with ASIL parameters which are severity for the damage and probability of apparition of hazard in a defined environment.

Rough dysfunctional scenarios can then be defined by instantiating for each flow deviation the diagram presented in figure 2. By analyzing these rough scenarios, we can identify critical ones and select them for further analysis.

By refining these critical scenarios, we can determine possible avoidance scenarios or safe states (nominal or degraded states of the system which are safe for the driver).

Finally, we have to conclude by determining a safety requirement that is useful to further design the system.

4.3 PHA Approach Related to Threats

The approach of Preliminary Hazard Analysis related to threats of the system on the environment is close to the global approach.

The main distinction concerns the determination of the hazard. It is not as systematic as in the global approach. Indeed, hazards due to threats are new abnormal flows that didn't exist before. For example, if we consider oil leak, there is no normal flow between vehicle and road.

5 Application to an Example

5.1 Determination of the Scenario

Diagram in Figure 5 identifies the flow between the vehicle and the road for the system service called "Accelerating". A deviation of the output flow is an "Unintended Acceleration". Now, we have to analyze whether there are potential damage due to this hazard.

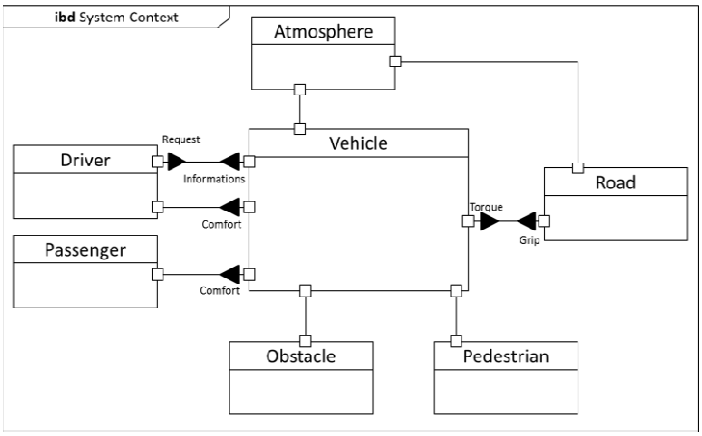


Fig. 4 Context Diagram

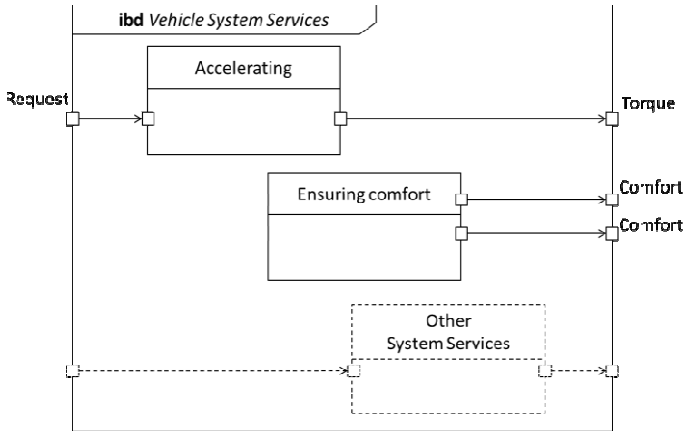


Fig. 5 Partial view of vehicle system functions

Thanks to the diagram on figure 4, we can determine that the worst case of damage is critical injuries of the passenger or pedestrian due to a collision.

So, we can have a scenario thanks to this information described as a diagram in Figure 6. We have environmental conditions favorable to hazard which is "presence of obstacle".

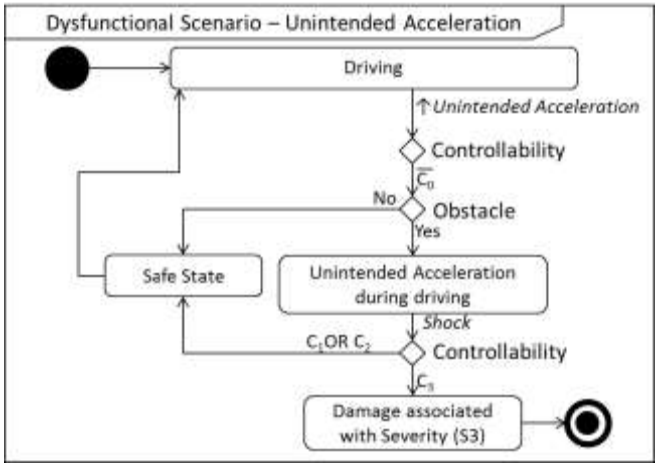


Fig. 6 Rough dysfunctional scenario of Unintended Acceleration

By analysing operational scenarios associated to the system service "Accelerating", this rough scenario can be linked to some operational scenarios. Moreover, we choose the one which has the worst consequences, i.e. critical injuries as we analyzed previously. In our case, the operational scenario is obstacle avoidance during driving in normal operation (Figure 7).

After short studies (like the determination of the perception time), we can modelize this dysfunctional scenario (Figure 8). A sequence diagram is chosen because we want to see interactions between the system and its environment.

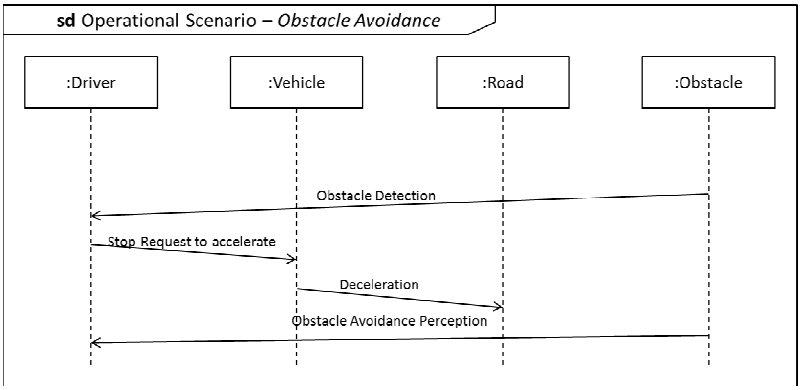


Fig. 7 Operational Scenario of Obstacle avoidance in normal operation

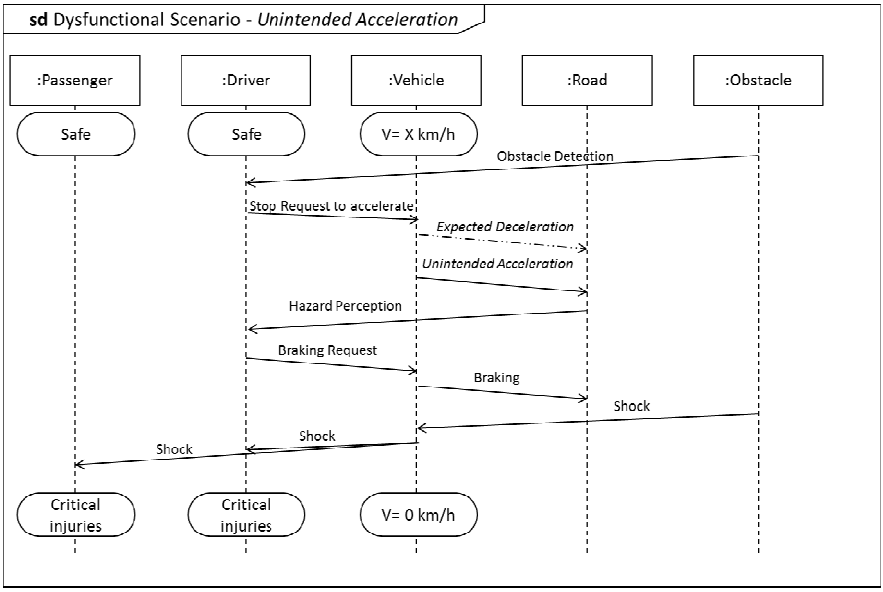


Fig. 8 Refined dysfunctional scenario of Unintended Acceleration

5.2 Assessment of the Dysfunctional Scenario (Example Used in ISO 26262)

Safety standards as ISO 26262 (3) contain tables to attribute a level for each criterion.

In our case study, driving is one of the basic phases of life cycle of the vehicle and accelerating is one of the main system services of the vehicle. So, thanks to the table of definition of this criterion in ISO 26262 (3), unintended acceleration during driving is E4 (E is for exposure and 4 is the highest exposure level).

Since the damage to the passengers are severe, the level for Severity is S3 (3 is the highest severity level) as it is defined in the table of severity in ISO 26262 (3).

Controllability is the most subjective criterion. It can be represented using the dysfunctional scenario modelized in figure 8. In our case study, we consider that 90 % or more of all drivers can react by braking if there is an unintended acceleration, so the level for Controllability is C2 (2 is the medium controllability level). ISO 26262 gives examples to define this criterion in a table.

5.3 Proposal of Risk Coverage

In the case study, the hazard corresponds to the output of the vehicle. So, possible avoidance scenario is to, either limit or reduce this performance. It leads to a safe state "without an unreasonable level of risk" (3).

5.4 Result of the PHA

As shown on table 2, the results can be synthesized for the corresponding Safety Goal. For reasons of readability in this article, it is presented into 2 parts. In the first one, there is the description of the dysfunctional scenario. In the second part, there is the assessment of the scenario and the proposal of risk coverage.

Table 2. PHA of "Unintended acceleration during driving"

System	Phase	Considered Flow	Hazard	Hazardous Event	Harmful Event	Operational Situation with Harm
Vehicle	Driving	Mechanical Energy to Road	Unintended Acceleration	Unintended Acceleration during driving	Shock	Critical Injuries of the driver

Probability of Exposure		Severity	Controllability	ASIL	Possible Avoidance Scenario	Safety Goal
X %	E4	S3	C2	C	Reduction of performances	the difference between driver's Intend and the acceleration of the vehicle must be less than Y%

6 Conclusion and Outlook

After having defined the various terms used to define a preliminary hazard analysis, we have defined a PHA process that is linked to the process of concept of operation and requirements analysis. Moreover, similar concepts have emerged from the conceptual model like mission and hazard. This emergence proves that systems engineering and safety activities can be done in osmosis and not separately like definition of operational and dysfunctional scenarios as it is shown in this article. The aim is to take into account the requirements of ISO 26262 and to better integrate MBSE and safety analysis.

The first outlook we will investigate is to link this work to the failure propagation in the functional architecture.

The second outlook focuses on the refinement of the process according to the de-sign stages (concept, preliminary design and detailed design). In the preliminary design, we have bases of system architecture. So global PHA can be done from the concept stage and refined in further ones.

Acknowledgments. We want to acknowledge PSA Peugeot Citroën and especially Dominique's Department for helping us to define this method and for the interesting discussions about this.

References

1. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety related systems (1998)
2. ARP 4761 - Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (1996)
3. ISO 26262: Functional safety for road vehicles (2009)
4. Desroches, A., Leroy, A., Vallée, F.: La gestion des risques: Principes et pratiques. Management et Informatique (2003)
5. Yakimets, N., Jaber, H., Lanusse, A.: Model-based system engineering for safety analysis of complex systems. In: MBSAW, Bordeaux (2012)
6. Papadopoulos, Y., McDermid, J.A.: Hierarchically Performed Hazard Origin and Propagation Studies. In: Felici, M., Kanoun, K., Pasquini, A. (eds.) SAFECOMP 1999. LNCS, vol. 1698, p. 139. Springer, Heidelberg (1999)
7. Batteux, M., Prosvirnova, T., Rauzy, A., Kloul, L.: The AltaRica 3.0 project for model-based safety assessment. In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN), pp. 741–746 (2013)
8. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An openSource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 359–364. Springer, Heidelberg (2002)
9. OMG Systems Modeling Language (OMG SysML) V1.3 (2012)
10. ISO 15288 Systems Engineering - System life cycle processes (2002)

11. David, P., Idasiak, V., Kratz, F.: Reliability study of complex physical systems using SysML. Reliability Engineering & System Safety 95(4), 431–450 (2010)
12. Cressent, R., David, P., Idasiak, V., Kratz, F.: Designing the database for a reliability aware Model-Based System Engineering process. In: Reliability Engineering and System Safety (2012)
13. Rausand, M.: Preliminary Hazard Analysis. In: System Reliability Theory. Wiley interscience (2004)
14. Chalé, H.G., Taofifenua, O., Gaudré, T., Topa, A., Lévy, N., Boulanger, J.-L.: Reducing the Gap Between Formal and Informal Worlds in Automotive Safety-Critical Systems. In: 21st Annual INCOSE International Symposium (2011)
15. Deniaud, S., Bonjour, E., Micaëlli, J.-P., Loise, D.: How to integrate reliability into Systems Engineering framework. A case from automotive industry. In: CRECOS seminar, Helsinki (2010)
16. IEEE 1220 - Standard for Application and Management of the Systems Engineering Process (2005)
17. Pyster, A., Olwell, D.H.: The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.1.2. The Trustees of the Stevens Institute of Technology, Hoboken (2013)

Taming the Complexity of Big Data Multi-cloud Applications with Models

Marcos Aurélio Almeida da Silva, Andrey Sadovykh,
Alessandra Bagnato, and Etienne Brosse

Abstract. Private and public clouds are getting more and more common. With them comes the need to analyse data stored by different applications in different clouds. Different clouds and applications tend to enforce the use of different data stores, which makes it even harder to aggregate information. The main outcome is that integrating different data sources requires deep knowledge on how data is stored on each solution and on the trade-offs involved in moving from one system to another. This paper is part of the ongoing work on the JUNIPER FP7 EU project. In that project we explore the power of modelling tools to simplify the design of industrial big data applications. In the present work we present an overview of our approach and its application on a simple case study.

Marcos Aurélio Almeida da Silva · Andrey Sadovykh ·
Alessandra Bagnato · Etienne Brosse
R&D Department, SOFTEAM, 9 Parc Ariane, Guyancourt, France
e-mail: {marcos.almeida,andrey.sadovykh,
alessandra.bagnato,etienne.brosse}@softeam.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_22

Scalability in System Design and Management, the MONDO Approach in an Industrial Project

Alessandra Bagnato, Etienne Brosse,
Marcos Aurélio Almeida da Silva, and Andrey Sadovykh

Abstract. The current system designs and management technologies are being stressed to their limits in terms of collaborative development, efficient management and persistence of large and complex models. As such, a new line of research is imperative in order to achieve scalability across the system design space. Scalability in system design has different dimensions: domains, team localizations, number of engineers, size and management of the engineering artefacts, interoperability and complexity of languages used. This paper depicts how the MONDO FP7 EU project (http://www.mondo-project.org/) aims to comprehensively tackle the challenge of scalability in system design and management by developing the theoretical foundations and an open-source implementation of a platform for scalable modelling and model management. An industrial case study is also presented. The system designed in this case study is distributed among several and dependent units, domains, and languages.

Alessandra Bagnato · Etienne Brosse · Marcos Aurélio Almeida da Silva · Andrey Sadovykh
R&D Department, SOFTEAM - 9 Parc Ariane, Guyancourt, France
e-mail: {marcos.almeida,andrey.sadovykh,
alessandra.bagnato,etienne.brosse}@softeam.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_23

Flexible Model-Based Simulation as a System's Design Driver

Jean-Philippe Schneider, Eric Senn, Joël Champeau, and Loïc Lagadec

Abstract. Complex systems traditionally involve partners from different companies with their own domains of expertise. During design stages, these partners need to exchange pieces of information and to debate around architectural and implementation choices. Model Driven Engineering for System Engineering simplifies system knowledge sharing, while simulation provides sound results to drive debate. As a consequence, gaining a flexible and dynamic tool that models and simulates the architecture is highly valuable. In this paper we focus on the functional architecture design and analysis steps of the system engineering process. We identify adaptation to existing system engineering process, tool modularity and interaction with models as three grounding principles for a flexible system model simulation tool. We show that meta-modeling and layered architecture for a simulator are enabling technologies for our three principles. We also demonstrate the use of these technologies by implementing a simulation tool in the context of a sea-floor observatory project.

Jean-Philippe Schneider · Eric Senn · Joël Champeau · Loïc Lagadec
UMR 6285, Lab-STICC, ENSTA Bretagne, 2 rue François Verny,
29806 Brest CEDEX 9, France
e-mail: {jean-philippe.schneider,joel.champeau,
loic.lagadec}@ensta-bretagne.fr

Eric Senn
UMR 6285, Lab-STICC, Université Bretagne Sud, Rue de Saint-Maudé, BP 92116 56321
Lorient CEDEX, France
e-mail: eric.senn@univ-ubs.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_24

Putting Real Production Software in the Loop, Methodologies Enabling SW Co-development between OEMs and Tier 1s

David Bailey, Gregory Nice, and Guillaume Francois

Abstract. With software gaining importance as the main contributor both to functionality and differentiation in the automotive market place and its relevance to quality, safety and customer satisfaction, its place in the development process and the methods available to ensure short development cycles and a simultaneously high level of quality are coming under strain. Both model-based and abstracted code – not specific to the final production target, are in use in the earlier phases but these often do not provide code which is testable in a meaningful way for the final product. In this paper we will explore methodologies which allow target independent code to be produced and managed as a product within the development process – establishing clear linkage between development code and the final product and accountability and traceability throughout the process. We will leverage the increasing implementation of Autosar and proliferation of model based sw development techniques in the process.

David Bailey · Gregory Nice
ETAS GmbH, Borsigstrasse 14, 70469 Stuttgart, Germany
e-mail: {David.bailey, Guillaume Francois}@etas.com

Guillaume Francois
ETAS France, 32, avenue Michelet, BP 170, 93404 St-Ouen – France
e-mail: Gregory.Nice@etas.com
www.etas.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_25

System Engineering, for a Cognitive Sciences Approach

Thomas Peugeot

Abstract. This article proposes to throw some light on the strong relationship between Cognitive Science and System Engineering. In the same manner that Herbert Simon justified hierarchical organizations with the argument that individuals had "bounded rationality", this article tries to show that most of our System Engineering processes are techniques to compensate for our cognitive boundaries. Practical implications for SE justification and SE education are discussed.

Thomas Peugeot
MOSS SAS, 86 rue Henri Farman, 92130 Issy Les Moulinaux, France
e-mail: thomas.peugeot@moss.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_26

Requirement Authoring Tools: Towards the Concept of Standard Requirement

José M. Fuentes, Anabel Fraga, Juan Llorens,
Gonzalo Génova, and Luis Alonso

Abstract. Several kinds of Requirements Management (RM) Tools are available in the market; it is difficult to know which one is the best for a project or a company. Guidelines regarding this task of selecting a tool and identifying the capabilities that can be desirable in the RM tools are offered among experts. After this long journey of guidelines, surveys and tools, many capabilities of the tools in the market have significantly evolved over the time, but they still lack other capabilities such as authoring assistant for system engineers when establishing standards for writing requirements. Most of the tools still show a blank textbox instead of an assistant for supporting engineers. A set of characteristics for writing good requirements using an authoring tools are shown and they are exemplified by requirements while presenting them. A comparison of tools is shown based on the desirable characteristics presented in this paper.

José M. Fuentes
The REUSE Company
e-mail: jose.fuentes@reusecompany.com

Luis Alonso
Carlos III University of Madrid
e-mail: {afraga,llorens,ggenova,luis.alonso}@inf.uc3m.es

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_27

iProd: Intelligent Management of Product Heterogeneous Data in the Product Development Process

Massimo D'Auria, Silvia Poles*, and Roberto d'Ippolito

Abstract. The product development process (PDP) in innovative companies is becoming more and more complex, encompassing many diverse activities, and involving a big number of actors, spread across different professions, teams and organizations. One of the major problems is that development activities usually depend on many different inputs and influencing factors, and that the information that is needed in order to make the best possible decisions is either not documented or embodied in data that is spread over many different IT-systems. Hence, a suitable knowledge management approach is required that must ensure the availability and the usability of required information. The authors suggest that this approach be based on a common semantic metamodel for representing engineering knowledge and that it should be applicable to companies from different industries. As a common reference model a generic, formalized PDP model is suggested that all required engineering knowledge can be associated with.

This paper presents results from the EC 7th Framework joint research project iProd. The project aims at improving the efficiency and quality of the Product Development Process (PDP) by providing an IT-supported knowledge management approach. In order to reach that goal, a flexible, service oriented, customer driven software framework is developed that applies semantic web, data integration and process integration and automation technologies and that 'maps' knowledge available in external systems (like PDM, requirements management systems, etc.) to a central knowledge metamodel, thus making it available in a semantically interpretable format without duplicating it. This will allow for

Massimo D'Auria · Silvia Poles · Roberto d'Ippolito
Noesis Solutions N.V., Gaston Geenslaan 11, B4 – B3001 Leuven, Belgium
e-mail: silvia.poles@gmail.com

* Corresponding author.

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_28

intelligently supporting knowledge-intensive engineering activities by providing context-based decision support. In order to showcase the potentials of this approach for Enterprise Integration and Interoperability, different real use cases have been identified together with the project's industrial end users and will serve as a basis for the development of demonstrators. In this paper one of these use cases is presented.

iProd addresses the PDP in a general way for manufacturing companies, but aims to prove the approach and methodologies in three well defined application areas, i.e., the aerospace, the automotive and the home appliances industries. These three areas generate the largest impact in European economy and are addressed here as the main targets for the iProd application.

Keywords: product development process, Enterprise systems integration and interoperability, process modeling, ontology.

Implementing ISO 26550:2013 Model Based

Andreas Korff

Abstract. Engineering complex systems on time and within budget now requires implementing the concept of re-use not only by "cloning and owning" low-level artifacts, but also by including the notion of product lines into a suitable development process. The new norm ISO 26550, published in late 2013, defines a reference model for Product Line Engineering and Management of Systems and Software, and is in-line with other relevant Systems and Software Engineering norms like ISO 15288 for systems or ISO 12207 for software.

We will describe the new norm, its important concepts and terms, and we will show in this paper that Product Lines and a model-based approach for Systems Engineering are together ideally suited for pragmatically implementing a PLE process according ISO 26550:2013.

Andreas Korff
Atego Systems GmbH, Major-Hirst-Str. 11, 38442 Wolfsburg, Germany
e-mail: andreas.korff@atego.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_29

Probabilistic System Summaries for Behavior Architecting

Michael Borth

Abstract. Smart system of systems adapt to their context, current situation, and configuration. To engineer such systems' behavior, we need to design and evaluate system-level control strategies and the intelligent management of key scenarios. We propose a model-based approach called *probabilistic system summaries* to explore related design choices, e.g., where to put the 'smarts' of a smart building. Our approach uses Bayesian inference to calculate effects of strategies and implementations, offering causal analysis of the costs and benefits of decision strategies in key scenarios. As the modeling is light-weight and suitable for various abstraction levels, probabilistic system summaries are appropriate for early but sound architecture decisions based on computational science. Next to its use within this analysis, the product of this engineering step, i.e., a Bayes net summarizing systems plus their environment, may form the core of decision making within the system of system.

Michael Borth
Embedded Systems Innovation by TNO, PO Box 6235, 5600 HE Eindhoven, NL
e-mail: Michael.Borth@tno.nl

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_30

www.UniText.fr
Information System Concept for Documentation/Project Management

Philippe Jarrin and Luc Beaupère

Abstract. The UniText tool is a concept in development that takes into account the distributed nature of documentation, which consists of multiple documents linked to several parts of the system described at different levels of abstraction, and produced by different people with different authority levels. UniText can manage the traceability of all the documentation units according to several criteria (author, date, context, tasks, sub-projects, delegation of projects), as well as suggestions and discussions about the documentation, in order to help building and maintaining a consistent documentation from all these pieces of information. The UniText venture is designed to structure the work of managers who are responsible for documentation in R&D projects, in order to improve clarity, motivation, efficiency, capabilities and to absorb and respect all contributions of all stakeholders.

Philippe Jarrin
e-mail: Philippe.Jarrin@Wanadoo.fr
www.UniText.fr

Luc Beaupère
e-mail: Luc.Beaupere@Magic.fr
www.LesAmisDeUniText.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_31

Correct by Prognosis: Methodology for a Contract-Based Refinement of Evolution Models

Christoph Etzien and Tayfun Gezgin

Abstract. The scope of this paper is collaborative, distributed safety-critical systems building up a larger scale system of systems (SoS). Systems are independently designed and can operate autonomously following both global SoS- and individual goals. A major aspect of SoSs is the evolution over time, i.e. the change of its architecture as a result of changes in the context of the SoS or the changes of individual or global goals. We define a modeling concept for evolution specifying all possible changes of the SoS over time. This evolution model is used to generate and analyze future architectures enabling the prediction of future violations of static specifications. The challenge is to address the consistency of the evolution model with respect to the static specification of the SoS. This is achieved by deriving so called dynamicity contracts and thus restricting the evolution model in such a manner, that only correct architectures are produced.

Christoph Etzien · Tayfun Gezgin
OFFIS – Institute for Information Technology,
Escherweg 2, 26121 Oldenburg, Germany
e-mail: {etzien,gezgin}@offis.de

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_32

Probabilistic Thinking to Support Early Evaluation of System Quality through Requirement Analysis

Mohammad Rajabalinejad* and Maarten G. Bonnema

Abstract. This paper focuses on coping with system quality in the early phases of design where there is lack of knowledge about a system, its functions or its architect. The paper encourages knowledge based evaluation of system quality and promotes probabilistic thinking. It states that probabilistic thinking facilitates communication between a system designer and other design stakeholders or specialists. It accommodates tolerance and flexibility in sharing opinions and embraces uncertain information. This uncertain information, however, is to be processed and combined. This study offers a basic framework to collect, process and combine uncertain information based on the probability theory. Our purpose is to offer a graphical tool used by a system designer, systems engineer or system architect for collecting information under uncertainty. An example shows the application of this method through a case study.

Keywords: system, quality, uncertainty, design, evaluation.

Mohammad Rajabalinejad · Maarten G. Bonnema
Faculty of Engineering Technology, University of Twente 2522 LW Enschede, Netherlands
e-mail: M.Rajabalinejad@utwente.nl

* Corresponding author.

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_33

System Engineering on 3DEXPERIENCE Platform – UAS Use Case

Frédéric Chauvin and Gauthier Fanmuy

Abstract. System Engineering has become increasingly complex over years. Not only the designed product itself must fulfill more requirements (functional, performances, ergonomics, interoperability, safety, …) but additional constraints such as environmental friendliness, globalization, OEM / Suppliers relationships, intellectual property, regulatory compliance and many more must be integrated in the design process. This implies an extended multidisciplinary collaboration between system engineers and beyond with all participants and stakeholder of the project.

The Unmanned Aerial System (UAS) use case illustrates how a modern tool suite can dramatically optimize the product development process. All the required information is made accessible in an integrated environment to ensure collaboration, project management, wise decision making and ultimately to help to imagine sustainable innovations capable of harmonizing product, nature and life.

Frédéric Chauvin
CATIA Multi-discipline & System R&D Director, Dassault Systèmes, 10, rue Marcel Dassault, 78140 Vélizy-Villacoublay, France

Gauthier Fanmuy
CATIA Systems Engineering User Experience Director, Dassault Systèmes, 10, rue Marcel Dassault, 78140 Vélizy-Villacoublay, France

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_34

Engineering a Parent-System, Designed to Generate Complex Sub-systems in the Field of Defense Planning

Wolfgang Peischel

Abstract. The main message and added value of this analysis lies in the development of a possible structure of defense planning, plus process logics, and in providing inherent functional principles of a "system-generating system" in the field of defense planning in a broader sense. This includes the identification of the interrelations of the researched basic principles.

The present research starts with the hypothesis that system development, particularly in highly aggregated social entities, is usually initiated by organizational structures that represent a complex system in themselves. This paper attempts to address the inherent functional logics of this specific "system of systems", which is the precondition for a successful development and for any control of systems and system-modeling per se.

The described system focuses on creation, control and further development of those sub-systems that provide an adequate reaction to existential threats and future challenges in unpredictable, uncertain situations. Functional principles of military system-planning will be deduced, analyzed, and presented in an abstraction that still allows for practical application in the private decision-making sector, as well. A possible civilian benefit might be gained, where these sets of skills (a specific military "unique selling proposition") are in demand.

Military system planning is based on specific functional principles that are tailored to leadership-decisions and system control in threatening, time-critical, and unforeseeable situations, usually in a volatile environment.

Attempting to explain according to which military scientific deductions a "system-generating system" in the area of defense planning could be developed, it will be shown in which areas military/leadership-science can offer research results also to civilian system development and where defense planning could benefit from other scientific branches.

Into the direction of private economy an insight is to be given, according to which system-logic military decisions are made respectively which basic principles guide planning-/ defense procurement-processes.

Wolfgang Peischel
National Defence Academy/Austrian Military Journal, 1070 Vienna, Stiftgasse 2a, Austria
e-mail: wolfgang.peischel@bmlvs.gv.at

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_35

Turning a Suite of Modeling and Processing Tools into a Production Grade System

Pascal Rivière and Olivier Rosec

Abstract. The French national agency for old age pensions has evolved, along a systemic vision of all the issues to be covered, a generic set of tools to design and process structured file formats. This set of tools has turned into a production grade modeling and validating suite which relies for its distribution on the continuous integration of its various components. As such, continuous integration has here turned into a system which industrializes the production of systems of tools, each characterized by the file format it implements and the version under which it is released.

Pascal Rivière and Olivier Rosec
CNAV, 110-112 av Flandre 75019 Paris, France
e-mail: {olivier.rosec,pascal.riviere}@cnav.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_36

A Geographic Information System Perspective for Large Scale Engineering Systems Design and Management – A Case Study for Sustainable Desalination Network in Saudi Arabia

Salma Aldawood, Abdulaziz Alhassan, Abdelkrim Doufene, Anas Alfaris, Adnan Alsaati, and Olivier de Weck

Abstract. We introduce a decision support system to evaluate and refine complex scenarios addressing the location, timing and technology of water and energy investments in Saudi Arabia. We start by giving insights on the utility of a geographic information system perspective. We explain the different layers we are

Salma Aldawood
Center for Complex Engineering System, King Abdulaziz City for Science and Technology
King Faisal Rd, Riyadh, Saudi Arabia
e-mail: s.aldawood@kacst.edu.sa

Abdulaziz Alhassan
Center for Complex Engineering System, King Abdulaziz City for Science and Technology
King Faisal Rd, Riyadh, Saudi Arabia
e-mail: a.alhassan@kacst.edu.sa

Abdelkrim Doufene
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge,
02139 MA, USA
e-mail: doufene@mit.edu

Anas Alfaris
Center for Complex Engineering System, King Abdulaziz City for Science and Technology
King Faisal Rd, Riyadh, Saudi Arabia
e-mail: a.alfaris@kacst.edu.sa

Adnan Alsaati
Center for Complex Engineering System, King Abdulaziz City for Science and Technology
King Faisal Rd, Riyadh, Saudi Arabia
e-mail: a.alsaati@kacst.edu.sa

Olivier de Weck
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge,
02139 MA, USA
e-mail: deweck@mit.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_37

addressing such as population, energy and water supply and demand over time, etc. We present the geospatial database model used for that purpose and an interactive website to support participative data collection and interactions with different stakeholders.

This work is part of a Strategic Sustainable Desalination Network project. The goal is to develop a platform for planning the efficient deployment of a sustainable desalination network in Saudi Arabia while promoting renewable resources. The platform takes into account key performance attributes such as sustainability, optimality, strategic security and robustness as well as the ideal phasing and deployment of the network.

If We Engineered Systems Like We Produce Movies

Dominique Luzeaux, Thierry Morlaye, and Jean-Luc Wippler

Abstract. If we engineered systems like we produce movies, would our enterprise and endeavors prove leaner and more agile? We daydream on the greener pastures of movie making, an industry whose signature practices were born out of necessity, to adapt to major regulatory upheavals in the 30's Hollywood, and have since proved remarkably effective & resilient. We journey through odd ways of tackling strangely familiar problems, subtly different organizational patterns, and seemingly unreasonable, yet strikingly efficient practices. Could we gain some insights or fresh ideas, to renew and better up our own 'complex system engineering' practices?

Dominique Luzeaux
Deputy Director, DIRISI, Ministry of Defense, France
e-mail: dominique.luzeaux@polytechnique.org

Thierry Morlaye
President, COERENSIS, Paris, France
e-mail: thierry.morlaye@coerensis.com

Jean-Luc Wippler
Owner, LUCA Ingénierie, Toulouse, France
e-mail: jlwippler@gmail.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_38

Interoperability between Design, Development, and Deployment of Safety – Critical Embedded Systems: CRYSTAL Project

Alexandre Ginisty, Frédérique Vallée, Elie Soubiran,
and Vidal-delmas Tchapet-Nya

Abstract. The overall goal of the ARTEMIS CRYSTAL project is to foster Europe's leading edge position in the design, development, and deployment of interoperable safety-critical embedded systems. Alstom Transport Use Case is focused on the development of a signaling system (especially on the function "Ensuring Safe Exchange in Station") and more precisely on interactions between transverse activities such as safety analyses, requirements management, traceability, and change management. Safety Architect is a CRYSTAL brick allowing to support local FMEA on the components of the system model and to automatically generate the Fault Trees. Using a system functional design or its physical architecture model, the user can perform a local analysis inside Safety Architect, by linking failure modes of the outputs of the components to the failure modes identified on the component inputs. Safety architect will provide the Traceability Relationships between the safety requirements, the system functions and the design requirements.

Alexandre Ginisty · Frédérique Vallée
All4tec, 91300, Massy, France
e-mail: {frederique.vallee,alexandre.ginisty}@all4tec.net

Elie Soubiran · Vidal-delmas Tchapet-Nya
IRT SystemX Integration Center Nano-Innov, Building N3 - 8,
avenue de la Vauve, 91120 Palaiseau, France
e-mail: {elie.soubiran,vidal-delmas.
tchapet-nya-ext}@transport.alstom.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_39

Using Orientor Theory for Coherent Decision Making for Application Landscape Design

Alexander W. Schneider and Florian Matthes

Abstract. More than 30 years have passed since the first enterprise architecture (EA) management framework has been published. While the field has reached a certain maturity in science and practice, a paradigm shift is just beginning. The emerging trend to regard an enterprise as complex adaptive system might allow redefining and achieving the holistic nature of EA management approaches. Thereby, the focus on static aspects of the architecture might be extended by taking behavioral aspects into account. In this paper, we argue (1) that currently available EA management approaches fall short in achieving the desired holism, i.e. viewing the system as a whole, (2) apply orientor theory to support decision making by deriving coordinated goals and principles for application landscape design and (3) show how a system-of-systems approach of applied orientor theory could look like. We conclude the paper by sketching out an appropriate research agenda.

Alexander W. Schneider · Florian Matthes
Technische Universität München, Boltzmannstr,
3, 85748 Garching bei München, Germany
e-mail: alexander.schneider@tum.de, matthes@in.tum.de

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_40

Reuse / Variability Management and System Engineering

Olivier Renault

Abstract: This paper aims to share industry experience in managing the reuse and variability in the industry and to analyze the linkage between this topic and system engineering. Reuse and variability management are two faces of the same coin and strongly influence business performance. "Hard" products industries (where mechanical and structural engineering were historically dominant) and "Soft" industries (where electronic and software engineering are dominant) addressed the questions from different perspectives. After describing the observed practices for managing the reuse and variability from the physical product standpoint, and taking in account concepts and approaches used in "Soft" industries, we analyze how systemic approach should help in better mastering the variability. In conclusion, we identify some principles and rules which would need to be investigated through research to better link PLM and systemic approach in variability management.

Olivier Renault
PLMSys Consulting, 111, avenue Victor Hugo, 75784 Paris Cedex 16, France
e-mail: olivier.renault.pro@gmail.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_41

Accounting for Uncertainty and Complexity in the Realization of Engineered Systems

Warren F. Smith, Jelena Milisavljevic, Maryam Sabeghi, Janet K. Allen, and Farrokh Mistree

Abstract. Industry is faced with complexity and uncertainty and we in academia are motivated to respond to these challenges. Hence this paper is the product of thoughts for exploring the model-based realization of engineered systems. From the perspective that the activity of designing is a decision making process, it follows that better decisions will be made when a decision maker is better informed about the available choices and the ramification of these choices. Presented in this paper, in the context of an example of designing a small thermal plant is a description of an approach to exploring the solution space in the process of designing complex systems and uncovering emergent properties. The question addressed is that given a relevant model, what new knowledge, understanding of emergent properties and insights can be gained by exercising the model? In this paper, the observations made are reported in the context of the Validation Square.

Keywords: Decision-based, Model-based, Compromise, Complex Systems, Solution Space Exploration, Decision Support Problem.

Warren F. Smith
School of Engineering and Information Technology, University of New South Wales, P.O. Box 7916, Canberra, BC ACT, 2610, Australia
e-mail: w.smith@adfa.edu.au

Jelena Milisavljevic · Maryam Sabeghi · Janet K. Allen · Farrokh Mistree
Systems Realization Laboratory, University of Oklahoma, 865 Asp Avenue, Felgar Hall, Norman, OK, 73019, USA
e-mail: {Jelena_84,maryam.sabeghi,janet.allen, farrokh.mistree}@ou.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_42

Complexity: Definition and Reduction Techniques Some Simple Thoughts on Complex Systems

Jon Wade and Babak Heydari

Abstract. Complexity can mean many things to many people. This paper presents an empirical, relativistic definition of complexity relating it to the system of interest, the behavior which is to be understood, and the capabilities of the viewer. A taxonomy of complexity is described based on this definition. Complexity and complication are compared and contrasted to provide some context for these definitions. Several methods of reducing or managing complexity are presented, namely abstraction, transformation, reduction and homogenization. Examples are given for each of these.

Keywords: System complexity, system complication, system complexity management.

Jon Wade · Babak Heydari
Stevens Institute of Technology, Castle Point on Hudson, Hoboken, New Jersey 07030, USA
e-mail: {jon.wade,babak.heydari}@stevens.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_43

Requirements for Single Pilot Operations in Commercial Aviation: A First High-level Cognitive Function Analysis

Guy André Boy

Abstract. Aeronautical engineering never stopped decreasing the number of technical crewmembers in commercial aircraft since the 1950s. Today, a new challenge has to be taken: single pilot operations (SPO). SPO consist of flying a commercial aircraft with only one technical crewmember in the cockpit, assisted by advanced onboard systems and ground operators providing flying support services. This next move is motivated by cost reduction, and must satisfy the same or better level of safety currently guaranteed with two-crewmen cockpit. This is a human-centered design (HCD) problem where decision-makers have to take risks. This paper presents an approach to risk taking in systems engineering. This approach is illustrated by the presentation of the difficult problem of SPO HCD, and the underlying function allocation problem.

Guy André Boy
Human-Centered Design Institute, Florida Institute of Technology,
150 West University Boulevard, FL 32901, U.S.A
e-mail: gboy@fit.edu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_44

Urban Lifecycle Management: System Architecture Applied to the Conception and Monitoring of Smart Cities

Claude Rochet and Florence Pinot de Villechenon

Abstract. At date, there is no standardized definition of what a smart city is, in spite many apply to propose a definition that fit with their offer, subsuming the whole of the city in one of its functions (smart grid, smart mobility…). Considering the smart cities as an ecosystem, that is to say a city that has systemic autopoeitic properties that are more than the sum of its parts, we develop an approach of modeling the smartness of the city. To understand how the city may behave as a sustainable ecosystem, we need a framework to design the interactions of the city subsystems. *First* we define a smart city as an ecosystem that is more than the sum of its parts, where sustainability is maintained through the interactions of urban functions. *Second*, we present a methodology to sustain the development over time of this ecosystem: Urban Lifecycle Management. *Third,* we define the tasks to be carried out by an integrator of the functions that constitute the smart city, we assume public administration has to play this role. *Fourth*, we present what should be a smart government for the smart city and the new capabilities to be developed.

Claude Rochet
Université d'Aix Marseille, France
e-mail: claude.rochet@univ-amu.fr

Florence Pinot de Villechenon
ESCP Europe, France
e-mail: pinot@escpeurope.eu

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_45

Toward Better Integration of Functional and Dysfunctional Models: Safety Architect

Frédérique Vallée, Anne-Catherine Vié, Jonathan Dumont, Nataliya Yakymets, Yupanqui Munoz Julho, and Agnès Lanusse

Abstract. As systems are becoming more complex, their safety assessment dramatically needs powerful tools. Most of the existing tools are poorly connected to the system design process and cannot be associated at early stages of development cycle. We introduce a model-based safety analysis (MBSA) methodology and its supporting tool: Safety Architect that permits better interactivity between design and safety assessment activities. A dysfunctional model is built from the system model described in SysML. It is used to specify possible failure-modes, mitigation barriers and propagation behavior at components level. From the specification of feared events (expressed in safety requirements), it can automatically produce propagation paths and highlight which components are potentially critical. Such critical paths related to feared events can be displayed on the system model for better understanding of failure sources. This cooperative safety analysis framework relies on the Papyrus modeling tool exploiting both its system modeling and advanced customization facilities.

Frédérique Vallée · Anne-Catherine Vié · Jonathan Dumont
ALL4TEC, 2-12 rue du chemin des Femmes, Odyssée E, 91300 Massy, France
e-mail: {frederique.vallee,anne-catherine.vie,
jonathan.dumont}@all4tec.net

Nataliya Yakymets · Yupanqui Munoz Julho · Agnès Lanusse
CEA Saclay Nano-INNOV, Institut CARNOT CEA LIST, DILS,PC174 and CEA, LIST, Laboratory of Model Driven Engineering for Embedded Systems, 91 191 Gif sur Yvette CEDEX, Saclay, France
e-mail: {nataliya.yakymets,yupanqui.munozjulho,
agnes.lanusse}@cea.fr

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_46

Active Experimentation and Computational Reflection for Design and Testing of Cyber-Physical Systems

Kirstie L. Bellman, Phyllis R. Nelson, and Christopher Landauer

Abstract. Cyber-physical systems are being deployed in a wide variety of applications, creating a highly-capable infrastructure of networked ``smart'' systems that utilize coordinated computational and physical resources to perform complicated tasks either autonomously or in cooperation with humans. The design and testing of these systems using current methods, while time-consuming and costly, is not necessarily sufficient to guarantee appropriate and trustworthy behavior, especially under unanticipated operational conditions. Biological systems offer possible examples of strategies for autonomous self-improvement, of which we explore one: active experimentation. The combined use of active experimentation driven by internal processes in the system itself and computational reflection (examining and modifying behavior and structure during operation) is proposed as an approach for developing trustworthy and adaptable complex systems. Examples are provided of implementation of these approaches in our CARS testbed. The potentialfor applying these approaches to improve the performance and trustworthyness of mission-critical systems of systems is explored.

Kirstie L. Bellman
Topcy House Consulting, Thousand Oaks CA, USA
e-mail: bellmanhome@yahoo.com

Phyllis R. Nelson
California State Polytechnic University Pomona, Pomona, CA 91768, USA
e-mail: prnelson@csupomona.edu

Christopher Landauer
Topcy House Consulting, Thousand Oaks CA, USA
e-mail: topcycal@gmail.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_47

Virtual and Physical Integration of Autonomous Vehicles for an Automated Humanitarian Mission in EasyChair

Pieter J. Mosterman, Justyna Zander, and Ascension Vizinho-Coutry

Abstract. As a recent report of the Intergovernmental Panel on Climate Change (1) confirmed, there's substantial evidence that global climate change exists. This change may well be responsible for intensifying effects of natural disasters such as storms, floods, earthquakes, and droughts that have an impact on the world population. With technology as a potential equalizer, here we explore requirements for humanitarian missions and the feasibility to address the natural disasters with emerging technologies and the cyber-physical systems paradigm (2, 3) tied to the human in the loop (4, 5). Our solution provides the survivors and the emergency personnel with information to locate and assist each other during a disaster event. The system allows to submit a help request to a MATLAB-based mission center connecting first responders, robots, drones, autonomous aircraft, and ground vehicles that are simulated with Simulink (6). The results are visualized in Google Earth interface.

References

1. IPCC Panel, The Fifth Assessment Report (AR5) of the United Nations Intergovernmental Panel on Climate Change (IPCC), Climate Change 2013: The Physical Science Basis, tech. report (2013)
2. Mosterman, P.J., Sanabria, D.E., Bilgin, E., Zhang, K., Zander, J.: A Heterogeneous Fleet of Vehicles for Automated Humanitarian Missions. Computing in Science and Engineering 16(3), 90–95 (2014)
3. Mosterman, P.J., Bilgin, E., Sanabria, D.E., Zhang, K., Zander, J.: Autonomous Vehicles on a Humanitarian Mission. In: SmartAmerica Challenge 2013 Workshop at White House, Washington, DC (2013)
4. Zander, J., Mosterman, P.J.: From Internet of Things through Computation of Things to a Prediction Engine. In: SmartAmerica Challenge 2013 Workshop at White House, Washington, DC (2013)
5. Zander, J., Mosterman, P.J.: Computation for Humanity—Information Technology to Advance Society. CRC Press/Taylor & Francis (October 2013) ISBN-10:1439883270
6. Smart Emergency Response System, `http://www.mathworks.com/smart`

Pieter J. Mosterman · Justyna Zander · Ascension Vizinho-Coutry
MATHWORKS
e-mail: {Justyna.Zander,avizinho,Pieter.Mosterman}@mathworks.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_48

Formal Framework for Ensuring Consistent System and Component Theories in the Design of Small Satellite

Jules Chenou, William Edmonson, Albert Esterline, and Natasha Neogi

Abstract. We present a design framework for small-satellite systems that ensures that (1) each satellite has a consistent theory to infer new information from information it perceives and (2) the theory for the entire system is consistent so that a satellite can infer new information from information communicated to it. This research contributes to our Reliable and Formal Design (RFD) process, which strives for designs that are "correct by construction" by introducing formal methods early. Our framework uses Barwise's channel theory, founded on category theory, and allied work in situation semantics and situation theory. Each satellite has a "classification", which consists of tokens (e.g., observed situations) and types (e.g., situation features) and a binary relation classifying tokens with types. The core of a system of classifications is a category-theoretic construct that amalgamates the several classifications. We show how to derive the theory associated with a classification and the theory of the system core, and we show how to check whether a given requirement is derivable from or consistent with a theory.

Jules Chenou · William Edmonson · Albert Esterline
NC A&T State University, 1601 East Market St., Greensboro, NC 27411, USA
e-mail: {jchenou,wwedmons,esterlin}@google.com

Natasha Neogi
National Institute of Aerospace, 100 Exploration Way, Hampton, VA 23666, USA
e-mail: neogi@nianet.org

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_49

SQUORE, an Actionable Dashboard Based on SE Leading Indicators to Manage System Engineering Effectiveness

Abstract. From basic measurement collection to Governance via Key Performance Indicators, the SQUORE actionable dashboard offers steering and dashboarding capabilities for:

- Software work products quality and standard compliance (MISRA, HIS, ISO9126, SQUARE 25010 …)
- System Engineering Effectiveness by collecting and publishing the trends of SE "Leading indicators" as defined by the INCOSE,
- Business process performance and support for compliance with International Standards such as Automotive Spice, CMMI, Six Sigma, …

Used in Systems Engineering context, SQUORE becomes the essential dashboard for SE Effectiveness monitorig by combining the following strategic capabilities:

- Aggregation of heterogeneous engineering data to get a comprehensive overview of products and processes: requirements, design models, test cases and results, documentation…
- Analysis models including a set of KPI specified in the INCOSE SE Leading Indicators Guide, allowing to follow the work progress and/or compliance all along the life cycle, periodically or at all key milestones (as in DoD 5000 or ISO/IEC 15288), and to compare products or projects performance.
- Reporting via a role-based dashboard with intuitive charts. The double data drill-down allows end users to explore either the hierarchy of work products generated all along the product life cycle to identify all non-compliant / under regression artefacts (e.g..: an ambiguous requirement, a complex design diagram, a unclosed change request,, a failed test case), or to drill down into the breakdown of Key Performance Indicators to assess "how a specific project activity is likely to affect system performance objective" (e.g..: how requirements volatility may affect functional suitability).
- Generating action plans according to predefined objectives and priorities using a user customizable trigger mechanism.
- Capitalization of data from past projects to correlate attributes (e.g. Test completeness) with external performance characteristics (e.g. Reliability) and provide predictive trend analysis to improve monitoring.

Squoring Technologies
76, allée Jean Jaurès, 31000 Toulouse, France
e-mail: contact@squoring.com

F. Boulanger et al. (eds.), *Complex Systems Design & Management*,
DOI: 10.1007/978-3-319-11617-4_50

Author Index

Abid, Manel 257
Agarwal, Siddhartha 175
Aldawood, Salma 341
Alfaris, Anas 341
Alhassan, Abdulaziz 341
Allen, Janet K. 351
Alonso, Luis 149, 321
Alsaati, Adnan 341
Arnaud, Mathilde 269
Almeida da Silva, Marcos Aurélio 311, 313

Bagnato, Alessandra 311, 313
Bailey, David 317
Bannour, Boutheina 269
Barais, Olivier 1
Beaupère, Luc 329
Becker, Johannes 15
Bellman, Kirstie L. 361
Bhatt, Antara 87
Bonjour, Eric 297
Bonnema, Maarten G. 333
Bonnissel, Marc 45
Borth, Michael 327
Boy, Guy André 355
Broodney, Henry 87
Brosse, Etienne 311, 313

Carlier, Aurélien 257
Champeau, Joël 315
Chapurlat, Vincent 225
Chatzimichailidou, Maria Mikela 105
Chauvin, Frédéric 335
Chenou, Jules 365
Collar Jr., Emilio 119
Combemale, Benoit 1
Costes, Joris 45
Couturier, Ludovic 257
Cuccuru, Arnaud 269

Dafaoui, El Mouloudi 59
Dagli, Cihan H. 175
Da Silva, Frédéric 257
D'Auria, Massimo 323
Degueule, Thomas 1
Deniaud, Samuel 297
de Villechenon, Florence Pinot 357
de Weck, Olivier 341
Dietz, J. Eric 207
d'Ippolito, Roberto 323
Dojutrek, Michelle S. 207
Dokas, Ioannis M. 105
Dony, Christophe 225
Doufene, Abdelkrim 341
Dumont, Jonathan 359

Edmonson, William 365
El Mhamedi, Abderrahman 59
Esterline, Albert 365
Etzien, Christoph 331

Fanmuy, Gauthier 335
Feldman, Yishai A. 73
Ferrogalini, Marco 239
Figay, Nicolas 59
Förg, Armin 15

Fraga, Anabel 149, 321
Francois, Guillaume 317
Fuentes, José M. 149, 321

Gaston, Christophe 269
Génova, Gonzalo 321
Gerard, Sebastien 269
Gezgin, Tayfun 331
Ghidaglia, Jean-Michel 45
Gilbert, Mark 15
Ginisty, Alexandre 345
Góngora, Hugo G. Chalé 239
Grolleau, Emmanuel 283

Heydari, Babak 353

Jarrin, Philippe 329
Jézéquel, Jean-Marc 1
Joannou, Demetrios 87
Julho, Yupanqui Munoz 359

Kalawsky, Roy 87
Kermad, Lyes 59
Kolski, Christophe 163
Korff, Andreas 325
Kreimeyer, Matthias 15

Labi, Samuel 207
Lagadec, Loïc 315
Landauer, Christopher 361
Lanusse, Agnès 359
Lapitre, Arnault 269
Levrat, Eric 297
Lienkamp, Markus 15
Llorens, Juan 149, 321
Loise, Dominique 297
Luzeaux, Dominique 133, 343

Maciá, Joaquim Lloveras 193
Madiot, Frédéric 283
Masagué, Sergio Gago 193
Masin, Michael 87
Matthes, Florian 347
Mauborgne, Pierre 297
Méndez-Acuña, David 1
Micaëlli, Jean-Pierre 297
Milisavljevic, Jelena 351
Mistree, Farrokh 351

Moones, Emna 59
Moreau, Christophe 239
Morlaye, Thierry 343
Mosterman, Pieter J. 363
Muguerra, Philippe 45
Munier-Kordon, Alix 257

Nastov, Blazo 225
Nelson, Phyllis R. 361
Neogi, Natasha 365
Nice, Gregory 317
Nielsen, Keld Lund 45
Nugent, Paul 119

Ouhammou, Yassine 283

Peischel, Wolfgang 337
Peugeot, Thomas 319
Pfister, François 225
Poirson, Benjamin 45
Poles, Silvia 323
Protopapas, Angelos 105

Raimbault, Juste 31
Rajabalinejad, Mohammad 333
Renault, Olivier 349
Rhodes, Donna 15
Richard, Michaël 283
Richard, Pascal 283
Riou, Xavier 45
Riviére, Pascal 339
Rochet, Claude 357
Rosec, Olivier 339
Ruault, Jean-René 163

Sabeghi, Maryam 351
Sadovykh, Andrey 311, 313
Sanduka, Imad 87
Saut, Jean-Philippe 45
Schneider, Alexander W. 347
Schneider, Jean-Philippe 315
Scremin, Lionel 257
Senn, Eric 315
Shani, Uri 87
Shindin, Evgeny 87
Smith, Warren F. 351
Soubiran, Elie 345
Stephan, François 59, 257

Tchapet-Nya, Vidal-delmas 345
Technologies, Squoring 367
Thoni, Olivier 257
Tian, Yingchun 87
Tschirhart, Fabien 257

Vallée, Frédérique 345, 359
Vanderhaegen, Frédéric 163
Vayatis, Nicolas 45
Vié, Anne-Catherine 359
Vizinho-Coutry, Ascension 363
Vosgien, Thomas 59

Wade, Jon 353
Wang, Renzhong 175
Wippler, Jean-Luc 343

Yakymets, Nataliya 359

Zander, Justyna 363