PLANNING AND ARCHITECTURAL DESIGN OF MODERN COMMAND CONTROL COMMUNICATIONS AND INFORMATION SYSTEMS

Military and Civilian Applications

THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE

PLANNING AND ARCHITECTURAL DESIGN OF MODERN COMMAND CONTROL COMMUNICATIONS AND INFORMATION SYSTEMS

Military and Civilian Applications

by

A. Nejat Ince
Istanbul Technical University Foundation
Centre for Defence Studies

Cem Evrendilek
Istanbul Technical University Foundation
Centre for Defence Studies

Dag Wilhelmsen
NATO C3 AGENCY, the Hague
The Netherlands

Fadıl Gezer
Istanbul Technical University Foundation
Centre for Defence Studies

Springer Science+Business Media, LLC

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

ISBN 978-1-4613-7823-5 ISBN 978-1-4615-6159-0 (eBook)
DOI 10.1007/978-1-4615-6159-0

TABLE OF CONTENTS

PREFACE

The subject of this book is Command Control Communication and Information (C^3I) which is the management infrastructure for any large or complex dynamic resource systems. Here command means the determination of what to do, and control means the ongoing management of the execution of a command.

Decision making is the essence of C^3I which is accomplished through a phased implementation of a set of facilities, communications, personnel, equipment and procedures for monitoring, forecasting, planning, directing, allocating resources, and generating options to achieve specific and general objectives.

The C^3I system that is in question here is for a strategic military command including its subordinate commands. Although the design methodology that will be expounded in the book is for a military system, it can, to a large extent, apply also to tactical military as well as to civilian management information systems (MIS).

A C^3I system is a decision making network that reflects a hierarchical organization of C^3I nodes. Each node is responsible for the management of some portion of the available resources, where the higher level nodes are responsible for a correspondingly greater portion of the resources. Within a C^3I system both command and control decision making occur at every level of the hierarchy. Command decisions at one level determine how to satisfy the management decisions at a higher level.

By taking similar national and international systems as a model, the book explains how the so-called management infrastructure can be planned and designed using available technologies, techniques and standards to achieve stated user operational requirements including time and cost constraints which may be imposed on the system.

The studies carried out in some national and international command headquarters show that the following user requirements and priorities may be given as generic requirements which can therefore be used as the basis for the general design work described in the book:

- C^3I system will generally consist of local and wide area nets, user terminals, servers, operating systems, databases and various application programs. Users through their terminals will access C^3I resources complying with security rules and other operational constrains.

- The system shall be used for usual C^3I functions as well as for training and exercises.

- The system architecture shall follow the recommendations and principles of the Open System Environment of ISO (International Standardization Organization). The system shall be scalable and extendable.

- Information exchange between commands shall be effected by means of message exchange, file exchange and data replication. Databases and application programs shall be distributed and commonly usable.

- The C^3I system shall function as a management information system in peace time and be ready to serve the command control needs in tension, crisis and war.

- The system shall be multi-level secure.

- The C^3I system infrastructure that is outlined above shall be equipped with software :

    a) to provide various "Common Applications" such as message handling, message processing, electronic mail, briefing support, etc.

    b) to provide "Functional Area Specific Applications" (Special Applications for intelligence, logistics, operations, etc.) these being also independent of the command level.

While Common Applications have a wide usage in all the commercial systems for which a large selection of COTS (Commercial Off-the Shelf) products is therefore available, there is a need to develop new software for Special Applications which are sometimes referred to as "computation-oriented applications" composed of various decision aids and support systems.

The book discusses the requirements above and the technical and operational issues raised and then deals with the main subject of designing C^3I systems.

In dealing with command and control, there appears to be four essential problems which must be overcome for a successful design:

1. There is no theory or generally accepted design methodology.

2. There is a wealth of data available in large-scale systems which are difficult to validate, access and correlate.

3. Progress in agreeing international standards often lags behind the relevant technologies which are developing very rapidly.

4. There are a number of systems implemented by different organizations with little or no coordination resulting in lack of interoperability, software portability and possibility of integration of facilities and resources.

It is believed that the book successfully tackles the above problems by employing an original design methodology based on the "mission-oriented analysis" of capturing user requirements from which system design parameters and functional areas to be automated are derived. A three-step approach is adopted to transform the requirements to a fielded system:

1. Transformation of strategic goals, generic characteristics and system specific characteristics into a Reference Model (RM).

2. Selection of standards and "standard" products for related functional elements in the RM to produce an architecture.

3. Selection of remaining products in accordance with the standards, set forth in the architecture, determining system performance and system integration requirements to provide a fieldable configuration.

An implementation plan can then be produced using technology and standards that are well understood and widely available. The system is developed through an evolutionary acquisition process where capability can be added incrementally as technology and standards progress and as applications software is created using prototyping where necessary.

The book contains specific information on Techniques and Technologies, which may be used for implementing a C^3I System:

- Mission oriented analysis methodology,
- Data modeling techniques
- Database technologies
- Communications
- MMI techniques
- Security techniques
- Software development and maintenance techniques
- Decision support systems and techniques
- Integration and interoperability techniques
- Survivability Design

The applicability of these mostly emerging techniques and technologies to C^3I systems is also evaluated in the book.

The subjects that are treated in the book are presented in the order which is usually followed by a system designer.

ACKNOWLEDGMENTS

Most of the material used in this book has come or derived from studies carried out under Prof İnce's direction in Turkey for "The Command Control Communications and Intelligence System" of The Turkish Armed Forces and benefitted from the work done over many years in NATO for The Supreme Allied Command Europe (SHAPE).

The authors are most grateful to many engineers and scientists who worked in the projects above therefore contributed indirectly to the book. It would be invidious to single out names but we would like to mention those who played important roles and to whom we owe thanks and appreciation: Mr. C. Büçkün, Mr. M. Seçgel and Mr. O. Çakıroğlu and our thanks and gratitude go also to the staff of Kluwer Academic Publishers as well as to Mr. N. Tufan, Mr. Z. Ener and Ms. K. Gül of The Centre For Defence Studies of İstanbul Technical University Foundation for the help given in the production of this book.

CHAPTER 1

INTRODUCTION

1.1. OBJECTIVE AND SCOPE

Several nations are planing and implementing military strategic and tactical Command, Control, Communications and Information (C^3I) systems. On the civilian side government and corporations are implementing Management Information Systems (MIS). The primary objective of this book is to give practical support to military and civilian system engineers and academics involved with planning, design and implementation of C^3I systems for military and civilian applications, so that they may be able to specify and implement fieldable Command Control and Management Information Systems which are directly derived from their operational requirements and which take into account the information systems technologies and standards that are available.

It is to be appreciated that the scale and complexity of C^3I systems are such that a comprehensive treatment of all the subjects and developments in C^3I would not be expected to be covered in one book. There are also some basic problems that make the adoption of an analytic approach difficult for system design. Nevertheless, the pragmatic approach taken in this book based on the experience of the authors over many years in designing and implementing many C^3I systems nationally and internationally would, it is hoped, ensure that those using the book would be able to plan and design large C^3I systems with available technologies, techniques and standards to meet user requirements ranging from data exchange, message handling, information processing and databases to communications and computer security, interoperability, survivability and decision support. Hence it will be seen that the book expounds an original design methodology based on the "mission-oriented analysis" of user requirements from which system design parameters and classification of functional areas for software development are derived.

The design process begins with a goal architecture including requirements baseline and main architectural principles in terms of generic, system-specific characteristics and strategic goals and continues with a three step approach, starting with the development of a reference model specifying functional elements of the system followed by the architectural design involving selection of standards for the functional elements and finally the specification of a fieldable configuration involving selection of hardware and software products compatible with the predetermined standards. An implementation plan, is then, produced by taking into account the priorities of requirements and various constraints such as available budget and time and finally a calculation of the cost of the system is presented.

A. N. Ince et al., *Planning and Architectural Design of Modern Command Control Communications and Information Systems* © Kluwer Academic Publishers 1997

1.1.1 Organizational Aspects

In this section, organizational structures of a military and civilian establishment are presented. The examples depicted for the two structures appear to be similar and it follows therefore that what is said and done for the military case would apply generaly to the civilian one.

A typical example of the structure of a military organization for which a C^3I system is to be designed, is shown in Figure 1.1.

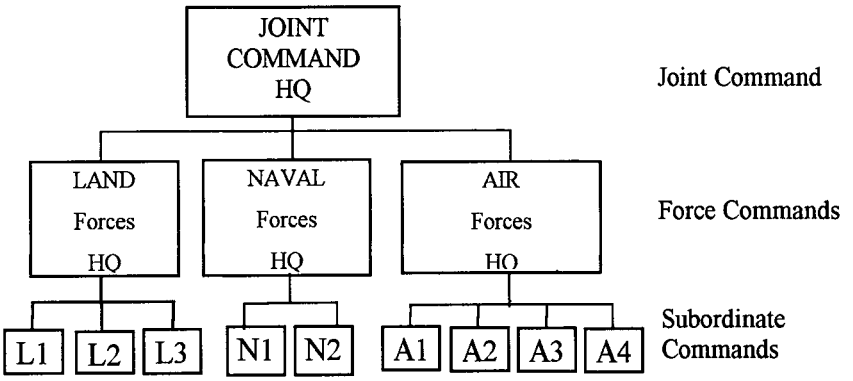


Figure 1.1 Military Organizational Structure

The military organization depicted in Figure 1.1, has a three level structure and at each level, there exists at the staff level the Divisions of Personnel, Intelligence, Operations, Logistics, Plans and Policies, Communications and Information Systems. These divisions are responsible for the conduct of principle activities related to:

- Command and Control,
- Communications and Information Systems (CIS),
- Exercises,
- Intelligence Activities,
- Logistics Planning and Execution,
- Crisis Management,
- Environmental Concerns,
- Electronic Warfare Operations,
- Force Planning,
- Budget & Finance,
- Personnel.

The requirements for a military C^3I system, while not identical, are not fundamentally different from that of a civilian commercial corporation. Rather than having functional divisions dealing with Sales, Production and Marketing, a military headquarters has divisions dealing with Operations, Intelligence, etc. In both military and civil organizations, there are departments dealing with Logistics

and Administration. In principle, therefore, the basic functions of an information system and its technical infrastructure in a military environment are the same as those in large civilian corporations, operating in the commercial environment. Indeed, the total amount of data that has to be processed in the latter is at least as large as in the former. MIS applications software needs are similar for both military and civilian organizations, whereas the military applications software - accounting for perhaps 10 % of the total Information System (IS) Software - naturally differs from the operational applications software of commercial operations.

This leads us to the conclusion that the Open Systems Architecture approach currently in general use and other general trends prevailing in such civilian information systems is equally applicable to military C^3I systems. Therefore, it would be a good starting point to briefly discuss what a civilian corporate organization demands and what technological trends are offered in response.

A similar corporate organization to that of the military organization with a three level structure is given in Figure 1.2.

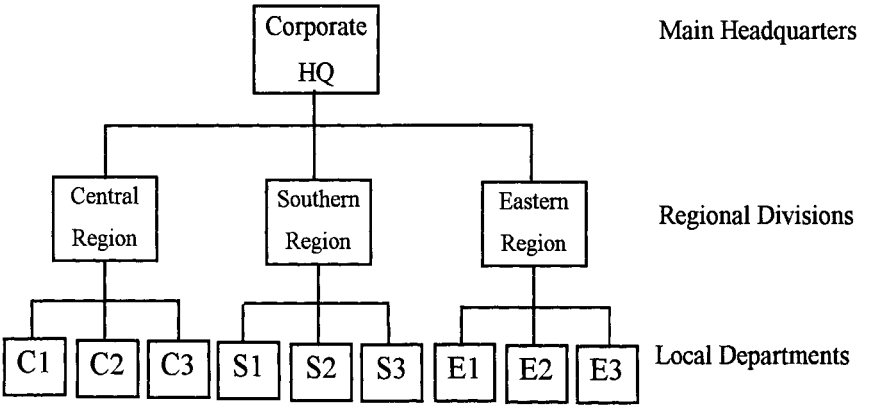


Figure 1.2 Corporate Structure

The main functions of this generic corporation are mainly related to:

- Executive,
- Sales,
- Marketing,
- Engineering R&D,
- Production,
- Logistics,
- Budget + Finance
- Administration,
- Personnel
- Service & Support,
- Projects,
- Human Resources.

The sole purpose of corporate information systems is to support the business processes that are required to carry out the company's mission. Corporate information technology (IT) infrastructures provide the information required to support and sustain these processes. This dictates a symbiotic relationship between IT and the other functions of the corporation, since IT's traditional administrative role has expanded to the point that it is pervasive in all aspects of managing the modern global enterprise.

One of the critical factors for a modern corporation's success is its ability to make and execute timely decisions, which, in turn, requires making widely available to all decision-makers timely and accurate information - such as market and competitive analysis, product and service development and delivery. In addition, as global markets evolve and business react with new corporate strategies and customer solutions, IT must be able to quickly change to provide the new information required to support the reengineered processes. Thus, the survival of the modern corporation depends on the ability of its IT architecture to respond to the constantly evolving information needs of its business processes. Clearly, business processes must not be driven by the capabilities of IT. Rather, IT must be ever responsive to the needs of the enterprise. Finally, current corporate trends include globalization, commodity product markets, shrinking market windows for the introduction of new products, a decrease in inventory cycles, and an orientation toward customer solutions with many heterogeneous products and services. More and more, decisions need to be made closer to the customer and in real time. It also is widely expected that management's span of control will greatly increase and that there will be a marked increase in the delegation of authority.

1.1.2 General System Description

By presenting a general description of the system to be designed, it is believed that the book will explain the design process more clearly and in a more illustrative manner by referring to and checking against this description where needed. The term "description" is used to denote the possible outlook after the design and implementation phases of the C^3I system are completed.

The current technology, standards and constraints are considered to be the key factors in the design for the realization of the description.

Needs of users, especially of decision makers including possible future demands are reflected in the description. The user needs are determined by structured knowledge elicitation techniques. Since user requirements are expected to change with time and these changes can be implemented as technology permits, the description may evolve incorporating new and enhanced features. This dynamically changing nature of the system is ensured in the design by employing the, so-called, evolutionary development and acquisition principle.

The following outline of a C^3I system is considered very typical for most applications of interest and will therefore be taken as the basis for design in this book:

- The topology of the organization for which the C^3I system is designed, is as given in Figure 1.1(See also Figs. 3.1 and 3.2).

- The system consists of a collection of processors, storage devices, terminals, and communication networks interconnecting them together with executive and application software.

- The system can be used for training and exercise purposes as well in addition to operational purposes.

- The system complies with the Open System Architecture principles. Thus, existing systems and systems procured in the future can be operated together enabling the system to evolve on demand smoothly. If all the system components conform to the standards, no technical interoperability problems arise. Software components can easily be transported from one platform to another. Problems of vendor dependence do not exist, because any vendor who supplies products conforming to the standards can be chosen. Therefore, in system purchase by evaluating more alternatives, procurement cost can be reduced.

- The system has a distributed structure and consists of a number of nodes located on geographically dispersed areas. Terminals are connected to each other with local and wide area networks. Each user location has local resources like networks, servers, databases. Data exchange among different locations can be performed by techniques like message exchange, and database replication. Consequently, all the command centers can share all the system resources.

- The system has redundancy for survivability to be used in peace, tension, crisis and war periods in the same or alternative headquarters.

- The system has the functionality of a Management Information System in peace and acts as a Command and Control Information System in tension, crisis and war.

- The main purpose of the system is to support decision makers by automation of their manual tasks. Here, supporting decision makers means aiding them by providing situation awareness, generating options, verifying and explaining the decisions in a timely and reliable manner and the system neither intends to replace human decision makers nor is this foreseen to be possible.

- The system contains measures to protect its users and their information against internal and external security threats attempting to actively and passively upset operations. Ideally the system should operate in accordance with Multi-Level Security policies. Thus, it could be possible to store, process, and transmit data of the highest security level in the system. However, the joint use of the resources are restricted for security reasons. Each user can access only the categories of resources permitted by the "need-to-know" principle and the clearance level of the user stating the classification level for the mission carried out.

1.2. DEFINITIONS

Definitions of the terminology used throughout the book are given below [1.1].

1.2.1. Command and Control (C2): The authority, responsibilities and activities of military commanders in the direction and co-ordination of military forces and in the implementation of orders related to the execution of operations.

1.2.2. Command and Control Communication System (C2CS): Communication system which conveys information between military authorities for command and control purposes.

1.2.3. Command and Control Information System (C2IS): Information system which provides military authorities with support for command and control purposes.

1.2.4. Command Control Communication and Information System (C^3IS): The term "Command Control Communication and Information System (C^3IS)" means " an integrated system comprised of doctrine, procedures, organizational structure, personnel, equipment, facilities and communications which provide authorities at all levels with timely and adequate data to plan, direct ad control their activities."

1.2.5. Commercial Off-The-Shelf (COTS): Pertaining to a commercially marketed product which normally is used without modification.

1.2.6. Communication: Information transfer according to agreed conventions.

1.2.7. Communications Security: Protection of information while it is being transmitted, particularly via telecommunications. A particular focus of communications security is protection of information confidentiality.

1.2.8. Communication System: Assembly of equipment[1], methods and procedures and, if necessary, personnel, organized to accomplish information transfer functions.

1.2.9. Computer Network: A network of data processing nodes that are interconnected for the purpose of data communication.

1.2.10. Computer Security: Protection of information while it is being processed or stored.

---

[1] A communication system provides communications between its users and may embrace transmission, switching subsystems and user /terminal area subsystem [1.2].

1.2.11. Configuration Management: The identification, control, accounting for, and auditing of all changes to system hardware, software, firmware, documentation, test plans, and test results throughout the development and operation of the system.

1.2.12. Common Open System Environment (COSE): It is a set of requirements that are used to derive a profile or selected list of specifications. These specifications define the environment in which the system functions.

1.2.13. Encryption: The transformation of original text (called plain text) into unintelligible text (called ciphertext). Sometimes called "enciphering."

1.2.14. Evolutionary Acquisition: Procurement of a broadly defined overall system capability in an adaptive and incremental way whereby the system is allowed to evolve[2] during planning, design and implementation within an approved architectural framework.

1.2.15. Information: Intelligence or knowledge capable of being represented in forms suitable for presentation, communication, storage or processing.

1.2.16. Information System: Assembly of equipment, methods and procedures and if necessary personnel, organized to accomplish information processing functions.

1.2.17. Management Information System (MIS): An information processing system that supports decision-making by the management of an organization.

1.2.18. Multi-Level Security (MLS): Multi-level security allows users at different sensitivity levels to access a system concurrently. The system permits each user to access only the data that he or she is authorized to access. A multi-level device is one on which a number of different levels of data can be processed.

1.2.19. Need-to-Know: A security principle stating that a user should have access only to the data he or she needs to perform a particular function.

1.2.20. Node: A system connected to a network.

1.2.21. Physical Architecture: The identification and arrangement of the physical components of a system architecture into an orderly framework that describes the physical structure, the technical functions, design features and technical attributes that can be achieved by each component and by the system within specified constraints.

---

[2] An underlying factor in evolutionary acquisition is the need to field a well defined core capability quickly in response to a validated requirement, including prototyping as appropriate, while planning through an incremental upgrade programme to eventually enhance the system to provide the overall system capability, including any required replication of system capabilities.

1.2.22. Prototype: A model or preliminary implementation suitable for evaluation of system design, performance and production potential; or for better understanding or determination of the requirements.

1.2.23. Rationalization: Method of satisfying the requirements of more parties by using resources, funded by the parties individually or in group, to constitute coherent, interoperable and cost-effective service capacities which are operated, managed and maintained under mutually agreed arrangements.

1.2.24. Software Development: Creation of new software implemented functionally to satisfy specified requirements.

1.2.25. Software Maintenance: Activity[3] intended to retain software in, or restore it to a state in which it can perform its required function.

1.2.26. System: Assembly of doctrines, methods, personnel, procedures, equipment or facilities organized to accomplish specific functions.

1.2.27. System Architecture: The logical structure and operating principles[4] of a system.

1.2.28. System Control: A function of system management for monitoring the status of the system and directing corrective action.

1.2.29. System Management: Planning, employment and control of system resources and their utilization to achieve and maintain optimal system operation.

1.2.30. Testbed: An environment containing the hardware, firmware, software, instrumentation tools, simulators and other support necessary for a representative testing and evaluation of a system or system element(s).

1.2.31. User: Any person or anything that uses the services of a communication system or information system.

1.2.32. Tempest: Collection of measures that prevents the compromising electrical and electromagnetic signals that emanate from computers and related equipment from being intercepted and deciphered.

---

[3] Software maintenance comprises corrective, adaptive, and perfective software maintenance.

[4] The operating principles include services, functions and interface standards against performance required and constraints imposed.

1.2.33. Vulnerability: A weakness in a computer system, or a point where the system is susceptible to attack. The weakness could be exploited to violate system security.

1.3. PROBLEM AREAS

The design of a C^3I system encompasses many difficulties [1.3] the sources of which may be classified as:

a) Uncertainty and changing nature of operational requirements to be obtained through knowledge elicitation techniques,
b) Lack of formal methods to express strategic goals to derive a goal architecture,
c) Realization of the intrinsic properties of a C^3I system stemming from the large and complex structure requiring an interdisciplinary study.

The third item, itself, may be broken down further into other problem domains such as;

- lack of a generally accepted foundation of C^2 theory,
- voluminous data to be processed rapidly and increasing complexity of modern warfare,
- the developments taking place rapidly in the electronics market place,
- loose coordination or sometimes no coordination at all in acquiring systems.

These problems, besides showing the difficulty of designing a C^3I system, also determine the main principle in design: Build a system which can easily and efficiently be scaled and extended incrementally on demand when it becomes feasible as permitted by the advances in technology and science.
To this end, some solutions to the aforementioned problems can be stated respectively as, use of formal methods in expressing and analyzing user requirements through computer aided software engineering tools; enumerating technical means in achieving strategic goals and imposing these means as standards throughout the design process; increasing the effort for validation and verification; extending software engineering techniques where possible or dividing the system to functional pieces of manageable sizes putting the emphasis on interfaces to design complex and large systems; to follow criteria for scalable and extendible systems; making use of efficient algorithms and fast computer architectures and determining issues in risk and uncertainty management; taking account of techniques for smooth interoperation of subsystems.

1.4. SYSTEM ACQUISITION PHASES

The system acquisition process for a C^3I system or indeed for any similar system, to be designed is divided into four phases [1.2]:

(i) **Establishment of the Concept:** Taking into account requirements, analysis, assumptions, state of technology and standards, objectives, design rules leading to system characteristics.

(ii) **Development of the Architecture:** Definition of the performance requirements of the subsystems and their integration in a manner which satisfies the level of performance required. This involves the study of each aspect of the concept in sufficient detail to produce costed proposals for budgetary cost estimates and to identify the major parameters and subsystem characteristics. This will, then, allow the project to pass into Phase (iii), which will include preparation of detailed cost estimates, technical specifications and implementation planning.

(iii) **Development of System Engineering Plans and Preparation of Specifications and Implementation Plan:** When approval is given from the Authority to the designer's proposals and recommendations developed in phase (ii) above, it will be possible to enter phase (iii) activities of specification writing and implementation planning (e.g. site surveys and plans). In this phase, the design authority will be required to produce a detailed cost estimate.

(iv) **Procurement, Installation and Commissioning:** The last phase of system acquisition when the equipment, software and services as specified in phase (ii) will be procured and installed according to the time phased plan and consistent with the operational requirements and funding constraints.

The design methodology adopted in this book covering steps (i) and (ii) of the system acquisition phases is presented below.

1.5. DESIGN METHODOLOGY

For any design authority that is tasked to design a C^3I type system for a specific organization, like a military organization in our case, we believe that the design methodology adapted would be similar to the approach reported in this book. The results and conclusions may vary because of different user requirements, constraints or available technologies, but the design approach described should be applicable in most cases. Some general comments should be made at this stage about the system we shall take as an example in this book and its characteristics. The system is defined as a strategic Command, Control, Communication and Information (C^3I) system to be owned and operated by its main users. The design and dimensioning for subordinate tactical C^3I systems is to be pursued in parallel.

The architecture for our system is influenced by several key operational user requirements, principally by the need for enhanced security and survivability, by the users to be served and the information to be exchanged. The detailed operational requirements must be developed as an iterative process involving both

the user/owner of the system as well as the design authority. This is to ensure that the requirements can be stated in terms which would lead to a feasible, cost-effective system capable of meeting the operational requirements. In order to develop the architecture, methodologies have been devised which enable the cost-effectiveness of the system in relation to the requirements to be assessed. The principle of this methodology as well as sequencing of the tasks are shown, in Figure 1.3.

Computer aided design tools are employed to dimension the system and to assess its performance for different configurations, conditions and under different scenarios of hostile actions from a potential opponent. The result of the studies is a set of costed options for implementing the system over a time scale consistent with the user requirements and within funding and other constraints.

A three-step approach is adopted for the design methodology covering phases (i) and (ii) of system acquisition:

i) Transformation of user-stated strategic goals, generic characteristics and system specific characteristics into a Reference Model (RM).

ii) Selection of standards and "standard" products for related functional elements in the RM to produce a System Level Architecture.

iii) Selection of remaining products in accordance with the standards, set forth in the architecture, determining system performance and system integration requirements to provide a fieldable configuration meeting the cost and other constraints given by the user.

These steps are expanded in the following sections.

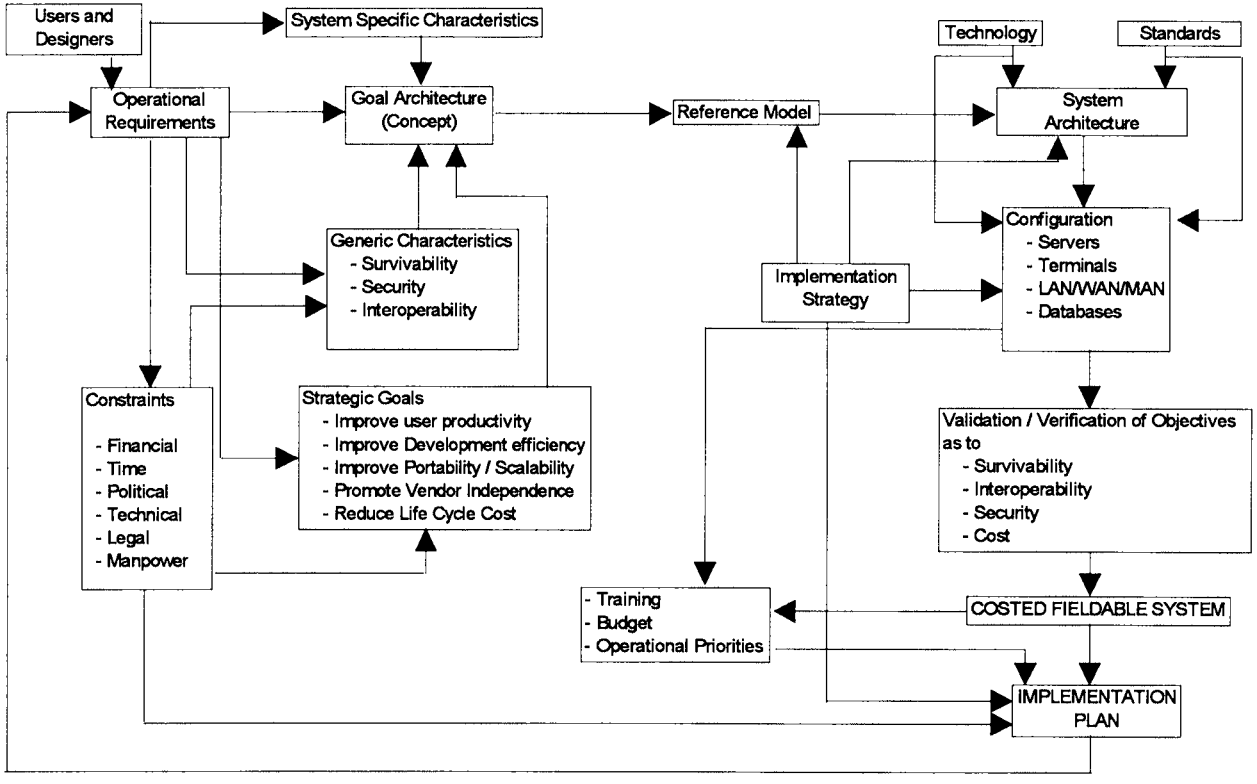


Figure 1.3 C^3I System Design Methodology

1.5.1 Concept

As can be seen from Figure 1.3, the main driver for the whole iterative process of designing the system is the "Concept".

Concept, sometimes called Goal Architecture, determines the infrastructure of the system through system specific characteristics, generic characteristics and strategic goals derived from the analysis of operational requirements [1.4].

Concept conveys information as to the user priorities, constraints to be imposed on the design and it serves both as a baseline and as a set of measures to check against validation and verification of the system design.

Strategic goals are the objectives of the system to be designed. They can be exemplified for the C^3I system to be designed as:

- Improve user productivity (through consistent user interfaces, integrated applications, data sharing, consistent security interface),
- Improve development efficiency (through common developments, COSE (Common Open Sytem Environment), COTS products, software reuse, resource sharing),
- Improve portability and scalability (portability, scalability),
- Improve interoperability (common infrastructure, standardization),
- Promote vendor independence (interchangeable components, non-proprietary specifications),
- Reduce life cycle cost (reduced duplication, reduced software maintenance costs, reduced training cost).

Generic characteristics of the system (Those that every system must have up to a degree) include features such as survivability, security and interoperability. Additionally, modern IT architectures must be:

Responsive: real-time information must be provided to support distributed decision-making.

Pervasive: all processes need to access and share the information necessary to support corporate strategies and tactics.

Distributed: distributed and mobile organizations will demand computing systems that are capable of being distributed and, thus, easily accessed locally.

Easily changed: future IT systems must be designed to evolve along with business processes.

Host-centered systems supporting off-site processes and heterogeneous information structures can not be quickly or easily modified to support emerging flexible organizations. It has been persuasively argued that traditional host-centered IT architectures are not optimal to support rapid change, distributed access, and data

conversion-making them unsuitable for the emerging global corporations of the 21st century. Thus, modern IT architectures must be, in contrast, network-centered.

System specific characteristics, on the other hand, are a result of functional requirements specific to functional areas.

In this phase of the design process, the system requirements are specified to derive later user-level functions, hardware and software configuration satisfying the constraints of performance, reliability, availability, serviceability and cost.

Goal Architecture containing also the topology of the system to be designed, is determined both with respect to organizational hierarchy and the activities performed in the organization.

As an example of topology formation, some properties for a C^3I system are given below showing the design decisions dictated and how a suitable topology is inferred:

Example

Property	Design Decision
diversity of activities	modular design
hierarchy in C2	centralized design
activity formation	client-server design

As depicted in the above table, diversity of activities in a C^3I system dictates a modular approach to design. Command and control structure implies a cenralized design approach. Formation of activities, on the other hand, may force the designer to adopt a strategy inclined towards client-server architecture.

The recent flood of articles and seminars about client-server computing indicates the growing popularity of this architecture for storing, manipulating, and viewing data. Its strength lies in its ability to distribute tasks, which gives the approach a booster shot of computing power. Neither the centralized computing of the 1970s nor the local computing of the '80s can compare for efficiency with this approach.

When a centralized system is used for a task such as database management, a single general-purpose mainframe or minicomputer handles every task except one- the display of the machine's output is left to attached terminals. In a local system used for such tasks, ordinary PCs manipulate as well as display stored data; the data may be stored on the local PC itself or a shared file server.

Client-server computing adds yet more intelligence, and specialized intelligence at that. In the interval between being stored on a file server or large computer and being finally processed and viewed on PCs, the data is preprocessed on an application server (either a specialized computer or software) dedicated to accessing and manipulating data. The application server is undoubtedly efficient. It sorts through large masses of data and passes only relevant information to the local computer. It is much faster than a large multi-user computer or a local PC at

juggling data because it is optimized for a single task. Since it transmits only relevant data, fewer bytes are moved over the network, thereby freeing up bandwidth.

In the traditional corporate computing environment, mainframe hardware and software systems were dominant. That is changing; today, corporations are shifting to distributed computing environment. Mainframe computing, also called host-centered computing, is based on proprietary and closed hardware and software, whereas distributed computing, also called network-centered computing, is based on open hardware and software.

New network-centered products and systems, unlike their host-centered counterparts, enjoy the technological and financial benefits of an active, global competitive market. In addition, the new distributed computing environments are better able to support current business needs, such as globalization and decentralized decision-making.

By design, distributed computing systems are more flexible and can be modified with minimum, localized impact, making them more responsive than host-centered systems to changing business processes and markets. On the other hand, the successful implementation of distributed systems poses new engineering and operational challenges because they are, by nature, a collection of independent resources using finite bandwidth networks to exchange information to collectively solve complex, data-intensive problems. Almost all of the improvements mentioned for civilian corporations on IT are applicable to military systems.

1.5.2 Reference Model

Given the strategic goals, generic and system specific characteristics; architectural elements, interfaces between functional elements and design characteristics are determined in the reference model [1.4].

Since the system to be designed is a C^3I system, the layers denoting the architectural elements of the system resemble to that of a typical information system such as hardware platform, operating system, and user interfaces.

The layers of the Reference Model (RM) should be complete in the sense that any entity or function required by the C^3I system to be designed, can easily be specified in terms of the elements of the layers unambiguously and easily.

The main components of the distributed computing architecture are;

- User interface,
- Applications,
- Application programming interfaces (APIs),
- Middleware (shared distributed computing services),
- System interfaces, and
- System and network infrastructure.

The basic principle of the architecture is the provisioning of as many computing shared services as possible in the form of shared servers, achieved through the middleware (shared distributed computing services) layer. Examples of shared services offered by this layer are directories, security, remote procedure calls, distributed time services, distributed file services, and data management and access. The architecture enables the logical integration of all system resources, makes resource location, access, and program execution transparent to the user, and supports mechanisms that enable efficient resource and system management.

The system and network infrastructure includes computing and storage resources interconnected via a network structure that may consist of a mixture of multiple access networks, switches, hubs, and internetworking devices.

Example

	Reference Model
1	Application Layer
2	API Layer
3	Common Services
4	OS
5	Hardware

Given the layers of the reference model as above, the communication subsystem would be realized as shown below in the corresponding layers:

1.	Audio/Video Information exchange
2.	Video Conferencing, E-Mail/Message Coordination, Secure-reliable image transfer, ...
3.	Routing, File Transfer Protocols, Naming, TCP/IP,
4.	Communication priorities of OS, Device Drivers, Threads, Message Passing, ...
5.	Communication Hardware (Routers, bridges), Interfaces, ...

1.5.3 Architecture

In this step, technical elements of the system are determined by populating the layers of the reference model with corresponding standards. These standards may be international, NATO or de-facto or some preferred standard products. Depending upon the concept, the use of these standards will be prioritized. For example, it may be the case that the strategic goal 'improve vendor independence' may dictate the use of international standards whereas, the generic characteristic 'security' may imply the use of some preferred standard product. This, in turn, requires to strike a balance between different design decisions without violating the overall concept.

It should be noted that, in this step, an evaluation of standards with respect to their maturity, functionality, quality and strength of service is required.

1.5.4 Configuration

Final step is the configuration step where physical elements of the system are determined and functional allocation is performed. The selection of remaining standards and standard products which result from the software to be developed and interfaces for interopeability, are realized in this step.

A combination of the dedicated-function and load-sharing is used for functional allocation. In the dedicated-function approach, the basic system functions are identified and assigned across multiple processors. This approach leads to simple and low-cost system architectures, but it may also lead to an unbalanced system, due either to an allocation imbalance or workload fluctuations.

In the load-sharing function approach, the workload is divided equally among a number of processors which, in aggregate, are capable of supporting the entire workload. The workload assignment process can be either static or dynamic. The use of both approaches permits the advantages of one to be used to offset the disadvantages of the other.

The major functions that need to be considered in the allocation process are:

- Electronic mail,
- File service,
- Database service,
- Computing service,
- Print service,
- Communication service,
- Data management services,
- Software distribution and management services,
- Distributed system management services, and
- Applications processing.

Hardware and software components making up the overall system are determined and the interfaces are specified in this step in such a way that the costed fieldable system implementation plan can readily be produced and validation and verification of the C^3I system designed, can be performed against the concept. This phase analyses access requirements and workload characteristics and uses analytical and simulation models to study system performance under various load conditions and evaluate design alternatives. The result is a system architecture that is then validated for its functional correctness and its ability to meet the design requirements. This step is iterative and may require several modifications until the design meets the objectives.

1.6 CONTENT OF THE BOOK

The subjects that are treated in the book are presented in the order which is usually followed by a system designer.

Chapter 1 starts with a summary of some problem areas and issues concerning C^3I systems and then outlines the phases of system acquisition followed by a description of system design methodology that is used in the book.

Chapter 2 deals with operational requirements; how they are captured and analyzed to derive system design parameters. Requirements are expressed in terms of command and conflict levels, key mission components, major C^2 activity categories and C^2 activities and decisions / actions for a military organization to establish a common dialogue between users and designers of the system.

User requirements are captured using three special forms for each command and conflict level which are named as,

- Form for Capturing Information Relating to KMC, C^2 Activities and Decisions / Actions,
- Form for Capturing Decisions / Actions Attributes,
- Data Set Form.

Techniques and procedures are then presented for requirements analysis, data and functional analysis and methods for derivation of system design parameters using Computer Aided System Engineering (CASE) tools.

In **Chapter 3,** the C^3I system to be designed is outlined in terms of command structure, network topology, information processing functions such as MMI, data management, information exchange, application software, computer security, system control and auditing, and communication subsystem and services (LAN/WAN/ATM techniques, Message Handling, File Transfer, Remote Data Access, Electronic Mail, Video) including communication security.

The C^3I system outlined is multi-level secure and is composed of a set of interconnected nodes each equipped with appropriate software and hardware facilities to serve the needs of users connected to that node. The nodes serving Main Headquarters are termed as primary nodes whereas nodes serving Subordinate Headquarters are called secondary nodes.

C^3I system supports Management Information System (MIS) activities as well as Command and Control functions. The system architecture adopted is based on client-server approach and conforms to agreed OSI protocols.

As mentioned previously system design follows a three-step approach, starting with the development of a reference model in terms of generic and system specific characteristics, and followed by the architectural design involving selection of standards and standard products and finally the specifications of a fieldable

configuration, involving selection of the remaining software and hardware products compatible with the selected standards.

In short, this chapter defines a goal architecture including requirements baseline and main architectural principles (generic characteristics, system specific characteristics, cost minimizing factors and strategic goals) which serves as a technical reference for the design and implementation of the C^3I system all of which are treated in the following chapters.

The objective of **Chapter 4** is to define a Reference Model (RM) that satisfies the goals and characteristics defined in previous chapter and which will facilitate the establishment of a specific C^3I architecture.

The Reference Model which organizes and interrelates all the technical components expected to be present in C^3I system is represented as a set of five interacting layers named:

Layer 1: Hardware Platform
Layer 2: Operating System
Layer 3: Common Services
(Interoperability, Data Management, Graphics,...)
Layer 4: Application
(Common User Functions, Special User Functions)
Layer 5: User Interface
(Character Mode, Block Mode, Graphical Mode)

As will be appreciated the purpose of RM is equivalent to the purpose of the OSI (Open System Interconnection) Reference Model. RM does not identify standards or products which are parts of the Chapter 5 and 6.

The design of the system technical architecture is dealt with in **Chapter 5** which basically describes how the layers of RM are populated with standards meeting the system characteristics and goals presented in the previous chapter.

The criteria used for selecting standards include considerations for security, availability of standards and COTS products, functionality, portability, scalability, interoperability and survivability.

All the available standards for different layers of the Reference Model are discussed which would give the system the attributes mentioned above. It is shown that not all the standards required are available as international ISO,IETF (Internet Exchange Technical Forum), CCITT (International Telegraph and Telephone Consultative Committee) or ETSI (European Telecommunication Standards Institute) standards and it is recommended here that where the international standards needed do not exist and the system implementation is urgent, use should be made of relevant NATO standards and emerging and /or temporary standards allowing the use of available COTS products that have been certified or authorized as providing adequate security protection.

In Chapter 6 the design of the C^3I system to be implemented is completed by specifying the hardware and software products which make up the system and comply with the standards set out in the previous chapter.

The hardware configuration embraces user terminals, servers, displays, LAN/MAN/WAN, bridges/routers, gateways, communication processors, and various security devices.

The software configuration consists of the programs for common applications such as message handling and the listing of functional area specific programs most of which are still to be developed using prototyping techniques. Taking account of operational requirements, these application programs are classified into three classes; database applications, graphics-oriented applications and computation-oriented applications.

The most onerous software development relate to computation-oriented software mostly decision support aids that require the use of techniques such as data fusion, AI (Artificial Intelligence), knowledge based expert systems, and black-boarding.

One of the most fundamental aspects of a military as well as of a civilian C^3I system is related to trust and security which covers three basic areas of confidentiality, integrity and availability of information and these are treated in **Chapter 7**.

These three topics are first discussed followed by a threat analysis including the vulnerabilities of the computer system in terms of personnel, physical, operational, communications, networks, computing and information. Possible security solutions are then presented in terms of authentication, access control, auditing and intrusion detection, trusted computing base (TCB) and network security leading to a multi-level secure system.

The extent to which the above solutions are implemented in the fielded system would determine the trustworthiness and security level of the system. The security level and the security risk, obtained by the system, can be estimated by a methodology outlined in this chapter. This methodology also establishes a relationship between user clearances and the assurance level of the hardware and software products used in the system.

The important subject of Surveillance and Control System (SURCONS) is treated in **Chapter 8, System Management,** where system features such as monitoring, control and maintenance, service and resource provisioning support to planning activities and system implementation and development are discussed.

SURCONS is designed to incorporate the above features with a view to maintaining maximum system performance as required by the user under changing operating conditions, natural and man-made stresses, disturbances and equipment disruption.

The SURCONS architecture supports centralized and decentralized modes of operation and reflects the distributed nature of the C^3I system and provides three

levels of monitoring and control performed at Major and Subordinate Headquarters.

Chapter 9, System Costing and Implementation, starts with the description of a costing methodology which is used to cost the C^3I system designed and which considers both the capital (one-time) and recurring costs (operation, maintenance, manpower and training) based on unit prices obtained from manufacturers and own experience capitalized over a number of years. Capital and recurring costs are tabulated in a typical format showing the expenditures on an annual basis for each related cost item.

This chapter then moves to discussing describes the essential features and strategy of implementation, taking account of the extent/coverage of the C^3I system, operational requirements and the uncertainties involved and financial and time constraints as well as problems raised by changing and evolving requirements and technologies.

Based on experience with military systems a step-by-step approach called "evolutionary development and acquisition" is proposed in this chapter which dictates an incremental acquisition process instead of implementing the overall C^3I system fully at once.

Adoption of a phased implementation strategy which affects implementation schedule is necessitated by factors such as, user priorities, status of technological advances and international standards, budget constraints, software development time and the number and distribution of nodes and installation times.

Chapter 9 recommends a 3-phase implementation schedule. The pressing user requirements related to Command Control information flow and some common user functions, so-called Core Capability, are implemented in Phase 1. This Phase provides also elements of the architecture (the infrastructure) which are essential to the evolution of the system towards the final configuration. Capabilities called Special Applications which are user specific are implemented in Phase 2. Some of the previous capabilities are extended and decision support which require relatively long time to prototype are implemented in the last phase.

The work involved in software development for C^3I is pioneering and experience is low, and yet it is important to produce realistic estimates of timescales, staffing and budget accurate enough for planning level cost estimates.

A methodology for estimating software development time and cost is presented in this chapter based on a mathematical model called "Quantitative Software Management" which requires an estimate of the amount of software to be written and an estimate of the productivity of the team that is to develop the software . Since neither of these parameters can be known accurately at an early stage of a novel project the resulting uncertainty is presented in terms of risk to the schedule, staffing and cost. The variability of the estimates produced in this way is reduced by the use of some historical database.

The book concludes with an **Epiloque** in which a synopsis is given of the important issues involved in designing and implementing a C^3I system incorporating new techological products and accommodating changing user requirements.

1.7 REFERENCES

[1.1] NATO CCIS Definitions & Glossary AAP-31(A), June 1993.

[1.2] Ince, A. N. et al, "Planning and Architectural Design of Integrated Services Digital Networks", Kluwer Academic Publishers, Boston, 1995.

[1.3] Harris, C. J., White, I., "Advances in Command Control & Communication Systems"", Peter Peragrinus Ltd., 1987.

[1.4] Hoegberg, K. T., "Recent Experiences Related to the Implementation of an Open System Architecture in Norwegian Forces", AFCEA Turkey Seminar, Ankara, 1993.

CHAPTER 2

METHODOLOGY FOR COLLECTING AND ANALYZING USER REQUIREMENTS
Mission-Oriented Analysis

2.1. COVERAGE

The purpose of this chapter is to present and describe a methodology for the elicitation and analysis of user requirements in designing an information system. For the reasons given in Chapter 1, the requirements capture will be for a military organization which we shall take as an example in this book.

It is a truism that the effectiveness and efficiency of an C^3I/MIS system to be designed for an organization will be as good as the quality of the user operational requirements captured and that this can be best achieved if a well thought-out methodology, easy to understand and use by the user and the designer, can be found or developed.

The methodology described and recommended in this Chapter for capturing and analysing user requirements is called "Mission Oriented Analysis (MOA) technique" which has been successfully used by the authors for national and international applications. This methodology has the merit of covering all the important issues affecting the system design and is claimed to be user friendly and in a format easily processable in a CASE (Computer Aided System Engineering) environment.

The categories of issues addressed and paid attention to hierarchically in the methodology for a military organization are given below:

The requirements capture is conducted in a hierarchical manner in that for each command level and conflict level (peace, tension, crisis and war time), the major C2 categories and activities are obtained for each categories of the key mission components of that command. This is then detailed by specifying decisions/actions and related data for each C2 activities.

Once the requirements are fully captured as above they are subjected to analysis using CASE tools from which the system design parameters are derived.

A. N. Ince et al., *Planning and Architectural Design of Modern Command Control Communications and Information Systems* © Kluwer Academic Publishers 1997

2.1.1 C^3I Applications for Military and Civilian Organizations

Command and Control (C2) is the process by which military commanders and civilian managers exercise authority and direction over their human and material resources to accomplish tactical and strategic objectives [2.1]. C2 is accomplished via the orchestrated implementation of a set of facilities, communications, personnel, equipment, and procedures for monitoring, forecasting, planning, directing, allocating resources, and generating options to achieve general and specific objectives. In civilian organizations, this process can best be called operational business management.

Military and civilian C^3I systems are similar in their requirements. Both military and civilian C^3I systems support the processes that are required to carry out the organization's mission. The success of both systems depends on their ability to make and execute timely decisions which, in turn, requires making widely available to all decision makers timely and accurate information. In industry, managers and corporate leaders identify market objectives and then mobilize resources to achieve them; in the military, commanders plan and execute complicated, phased operations to fulfill their missions. Managers in industry mobilize factories, aggressive managers, line workers, and their natural and synthesized resources to produce superior products. Commanders in the military mobilize weapons, troops, and sophisticated communications equipment to defend and acquire territory and associated military and political objectives.

The collection of user requirements is key to the successful design of any information system as well as C^3I systems. The design methodology described in this book heavily relies on this phase requiring a very careful, unambigious, correct understanding of what the users of the C^3I system are going to perform and what are the processes involved and the data to be processed. In this respect, the requirements capture is relatively easier in military than in civilian organizations since the description and details of most of the military activities, are well regulated and defined in directives in terms of command and conflict levels, key missions and categories. This implies that the Mission Oriented Analysis (MOA) methodology described in this chapter, when applied to a civilian organization, will require greater effort from the user. However, once the requirements are captured in the right and correct way following the MOA methodology, the analysis of the data gathered can then follow strict and auditable procedures leading to better designed systems.

The MOA methodology allows incorporation of criteria for security and survivability demanded by military systems. Therefore, the designers of C^3I systems for civilian corporations who have already started to pay more and more attention to security and survivability characteristics, should find MOA a useful and effective methodology for capturing requirements data.

2.2. ELEMENTS OF MOA

2.2.1 Command Levels

As mentioned in Chapter 1, this book describes the design of a C^3I system taking the military organization as an exampe. A generic organizational structure for a military organization is to be illustrated is given in Figure 1.1 of Chapter 1. This 3-layered organizational structure depicts a hierarchy whose levels correspond to Joint Command (JC), Force Commands (FC) and their Subordinate Commands (SC) from top to bottom. Each command has both a peace and generally a separate protected war headquarters for conducting peace and war time activities. The information needed by the Joint Command is usually obtained from Force Commands in peace, tension, crisis and war, but in war conditions, the required information may also be attained directly from Subordinate Commands in addition to FCs. The decisions produced in response are instantly delivered to subordinate headquarters and the results of actions correcponding to these decisions are monitored continuously through completed activity reports sent by subordinate commands. In the light of the explanations given briefly, the headquarters that will be served by a C^3I system can be listed as follows:

- Joint Command Peace and War Headquarters (PHQ, WHQ),
- Force Commands Peace and War Headquarters,
- Subordinate Commands Peace and War Headquarters.

In some cases, JC and FC war headquarters may be co-located in order to increase the effectiveness in administration and tasking of the forces and for better collaboration and coordination.

As it would be expected and as seen in Figure 1.2, the organizational structure of large civilian corporations resembles very closely that of the military organization in peace.

2.2.2 Conflict Levels

The main objective of the military activities taking place in a headquarters is to enable the commander to take appropriate and timely actions in response to any conflict situation including war that may develop. It would be expected that the information needed by the commander would change as the situation changes from peace to war time conditions. In order to be able to design the system to have the characteristics which can accommodate the changes in question it is necessary to elicit the commander's (user's) requirements for each discernable conflict level. Three conflict levels as listed below are defined for command and control purposes:

- Peace,
- Tension/Crisis,
- War.

The functional requirements and information needs for these functions are determined separately for each conflict level.

2.2.3 Key Mission Components (KMC)

For each command level and consistent with its strategic objectives the user specify what are his key missions; we call these the key mission components of the Command. Typically a KMC may be as given below:

(i) Peace time missions,
(ii) Crisis management,
(iii) Prevention of initial attack,
(iv) Air superiority and sustenance,
(v) Prevention of enemy's main attack,
(vi) Sea control,
(vii) General counter attack.

For each KMC, the user must specify the major categories of C2 activities to be undertaken by his command in a given conflict situation. Corresponding to each C2 category the user must then specify the C2 activities involved. Finally for each C2 activity the user must specify what decisions are to be made and the actions to be taken by his command.

C2 activities and decisions/actions in peace time in contrast to the activities in other conflict levels which are formed individually, are determined as dictated or required by the other conflict levels. Changes in threat may in parallel bring about new key mission activities or may result in alteration of key mission activities. This, in turn, necessitates the infrastructure of the C^3I system to be so designed so as to present facilities for realizing such possible additions and updates.

Key mission activities are dependent on the conflict levels. Not all the key mission activities can be performed in every conflict level. Military activities in support of crisis management, for example, are performed only in crisis. Whereas, key missions related to activities such as prevention of initial attack, air superiority and prevention of enemy's main attack are only executed in war time but not performed in peace. Sea control, on the other hand, is a key mission activity required in every conflict level. The relationship between the key mission activities and the conflict levels are shown in Table 2.1 below.

Table 2.1 Key Mission Components and Conflict Levels

KEY MISSION COMPONENTS	CONFLICT LEVELS			
	Peace	Tension/ Crisis	Conventional War	Nuclear War
Peace Time Missions	+	+	+	+
Crisis Management		+		
Prevention of Initial Attack			+	
Air Superiority			+	
Prevention of Enemy's Main Attack			+	
Sea Control		+	+	+
General Counter Attack			+	+

The information as outlined above can be collected using the form given in Table 2.2 where, for the information sought under the headings of C2 categories, C2 activities, decisions/actions, and data bases, typical examples are provided in Section 2.2.4, 2.2.5 and 2.2.6 below.

2.2.4 Command And Control Categories

C2 activities for accomplishing key missions are divided into seven categories as shown in Figure 2.1.

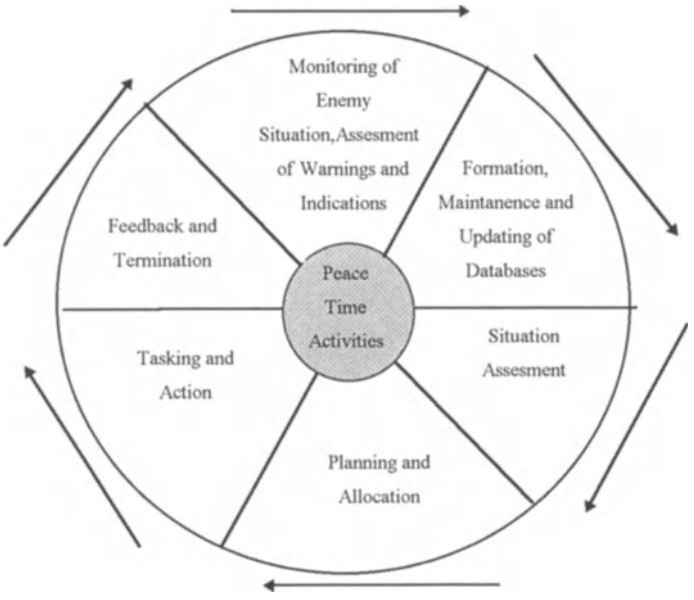


Figure 2.1 Typical Command -Control Activity Categories

(i) Peace time activities,
(ii) Monitoring of enemy situation and assessment of warnings and indications,
(iii) Formation, maintenance and update of related databases,
(iv) Situation assessment,
(v) Planning and allocation,
(vi) Tasking and action,
(vii) Feedback and termination.

2.2.5 Command and Control Activities

Typical Command and Control Activities performed during different conflict levels in various C2 Categories are enumerated in the following:

- Collection and distribution of intelligence data,
- Operation of sensors and warning systems,
- Coordination of reconnaissance and surveillance,
- Utilization of intelligence databases,
- Utilization of operations databases,
- Utilization of logistics databases,
- Utilization of communications and information systems databases,
- Utilization of personnel databases,
- Utilization of targets databases,
- Threat assessment,
- Revision of reserve requirements,
- Operations Planning,
- Revision of constraints,
- Notification of military vigilance,
- Notification and employment of alert measures,
- Psychological war activities,
- Deployment of forces,
- Application of civilian military cooperation,
- Revision of security requirements,
- Tasking and action,
- Termination of vigilance,
- Termination and reporting.

In determining the C2 Activities, the following assumptions are taken into consideration:

- Activities, decisions and actions carried out in the JC Headquarters are taken as basis. The data that can support these decisions and actions, however, will be acquired from subordinate commands (FC and SC) in addition to the JC Headquarters. The decisions taken by the JC Headquarters will be delivered to the subordinate commands and the results of the operations will be fed back to the JC.

- Some key missions will require consultation to political authorities and the decisions of these authorities.

- Land, air and naval operations can be carried out independently or jointly.

- Key missions can be conducted jointly depending on the scenario.

2.2.6 Decisions/Actions And Data Bases

2.2.6.1 Decisions/Actions

Decisions/actions necessary for each C2 activity together with associated data sets and functional areas and commands are listed in the last columns of Table 2.2.

The functional areas in a command typically can be Personnel (F1), Intelligence (F2), Operations (F3), Logistics (F4), Plans and Policies (F5) and Communications and Information Systems (F6).

Table 2.3 Form For Capturing Decision/Action Properties

DECISION/ACTION	
EXECUTING AUTHORITY	
COORDINATION AUTHORITY	
RELATED DOCUMENTS	

DATA PROCESSING REQUIREMENTS

voice/image processing	message preparation	message processing	word processing	graphics	briefing	GIS	electronic tabulation
voice/image output presentation	Digital Processing	Large Screen Display		Image	Voice		

SECURITY LEVEL

Unclassified	Restricted	Confidential	Secret	Top Secret	Cosmic	Atomal

SECURITY MODE

MLS	**CMW**	**SH**	**Dedicated**

PRIORITY/SIGNIFICANCE	
PREREQISITE DECISIONS/ACTIONS	
AFFECTED DECISIONS/ ACTIONS	

PROBLEM AREAS

Table 2.2 Form For Capturing Information Relating to KMC, C2 Activities and Decisions / Actions

Key Mission Component :
Conflict Level :
Command Level :

Major C2 Catagories	Code No.	C2 Activities	Code No.	Decision / Action	Code No.	Data Set		Elements of Data Set		Related Decisions/ Actions	USERS							
											FUNCTIONAL AREAS				OTHER COMMANDS			
						Name	Code	Name	Code		F1	F2	...	F6				

A) For each decision/action the following information will be collected (Table 2.3).

i) **Executing Authority**
ii) **Coordinating Authority**

If the executing and the coordinating authorities are in JC the names of related branches, otherwise the names of related FC and SC are given.

iii) **Data Processing Requirements**

The data processing requirements for decisions/actions are given in terms of specifications listed below:

- Voice Recognition: Digitizing voice signal, analyzing, identifying, converting into commands and text, and then storing, etc.

- Image Processing: Analyzing an image data (e.g. air photographs), identifying, and storing, etc.

- Message Preparation: Converting data into formatted messages, and augmenting data like destinations, prorities, security levels of the messages and then transferring to message handling system, etc.

- Message Processing: Updating the related databases using the data of formatted messages received from message handling system.

- Word Processing: Commercial software like Word, Word Star, Word Perfect, etc.

  Graphics: Software and hardware (graphics screen, plotter, etc) required to create and to wiew graphics drawings.

- Briefing Support: Tools for faster, trusted and confidential briefing preparation and presentation.

- Geographical Information System (GIS): The need for software enabling for instance personnel and logistics data to be viewed on maps; supporting interactions between GIS and DBMS in such a way that updates in one are automatically reflected in the other etc.

- Spreadsheet: Need for spreadsheets.

- Number Crunching: Need for software and hardware for speeding up compute-intensive tasks.

- Large Screen: Electronic map display for briefing.

iv) **Security Level**

The security level of decisions/actions is given by the following classification:

- Unclassified
- Restricted
- Confidential
- Secret
- Top Secret
- Cosmic
- Atomal

v) **Security Mode**

The security mode applied by the executing and coordinating authorities can be one of the following:

- Multi Level Security (MLS)
- Compartmented Mode Security (CMS)
- System High (SH)
- Dedicated

vi) **Priority/Significance**

The priority/significance of decisions/actions determines how the detection and counter measures are effected in case of performance-degradation case.

vii) **Prerequisite Decisions/Actions**

Previous set of decisions/actions affecting the decisions/actions in question.

viii) **Affected Decisions/Actions**

The decisions/actions in question affecting other decisions/actions.

ix) **Problem Areas**

Organizational, bureaucratic problems encountered in executing actions/decisions some of which may be solved by automation.

One of the major aims of the C^3I system to be designed is to aid the decision makers by producing decision alternatives, and to dynamically assess alternatives on the basis of evaluation criteria. This, on the other hand, may not be possible to do automatically with the current technology and knowledge or it may not even be required for some cases. For this reason, it is necessary to determine the automation level of each command and control activity. The data required for the actions and decisions which are to be taken in response to different events will be accessed from

the databases which always contain up-to-date information. Since data can reside in different systems in different headquarters and may have different security levels, interoperability and multilevel security issues should be taken into consideration. The issue of distributed or centralized use of databases in geographically dispersed headquarters should address both availability and survivability in providing for a solution.

B) For each data item the follwing information will be provided (Table 2.4).

i) **Name of data**
ii) **Source and destination of data**
iii) **Owner of data**
iv) **Branch Number**

The branch in JC which is the source, owner or destination of data.

v) **Validation**

If data is validated, the method and the sources used for validation.

vi) **Type**

The type of information is given by the following:

- Formatted Text (message)
- Unformatted Text
- Voice
- Photo
- Video

vii) **Amount/Frequency**

The amount of data to be given in number of pages. The frequency of data is given by:

- Hours
- Days
- Weeks
- Months
- Years
- Once

Table 2.4 Data Set Form

DATA SET NAME	
DATA SET	

Source & Destination	Branch No	Validation		Type					Amount / Frequency							Security Level							Security Mode					FORMAT		
		how	no	message	text	voice	photo	video	page	day	week	month	3 months	year	once	unclassified	restricted	confidential	secret	top secret	cosmic	atomal	MLS	CMS	SH	Dedicated	PRIORITIES	ADATP3	ARMED FORCES	ACP127
FROM 1																														
FROM 2																														
FROM 3																														
FROM 4																														
FROM 5																														
TO 1																														
TO 2																														
TO 3																														
TO 4																														
TO 5																														
OWNER																														

viii) **Security Level**

The security level of data is given by the following classifications marks:

- Unclassified
- Restricted
- Confidential
- Secret
- Top Secret
- Cosmic
- Atomal

ix) **Security Mode**

The security mode to be used in processing and flow of data can be one of the following:

- Multi Level Security (MLS)
- Compartmented Mode Security (CMS)
- System High (SH)
- Dedicated

x) **Precedence/Priority**

This can be one of the following categories:

- Flash (P1),
- Immediate (P2),
- Priority (P3),
- Routine (P4).

xi) **Message Formats**

- ADAT-P3,
- National standards,
- ACP-127.

2.2.6.2 Databases

Conducting Command and Control activities timely, thoroughly and correctly will only be possible if the information needed can be acquired, stored and processed correctly and quickly. This, in turn, requires special attention to be paid to the selection of a Data Base Management System (DBMS) to be used and the databases to be created.

The databases that are used in carrying out command and control activities and the information which can be accessed through these databases are created to correspond to the functional areas which are common to all the commands. The important data elements contained in each data base are given below.

a) Personnel

This database contains information about personnel assesment such as:

- Personnel posts/availability,
- General personnel availability,
- Critical personnel situation,
- Personnel education/training status,
- General readiness status.

b) Intelligence

- Intelligence outlines, reports and complementary reports from a variety of sources,
- Information regarding the enemy's combat establishment and deployment,
- Background studies for regions of operations,
- Intelligence situation assessment,
- Intelligence countermeasures areas,
- Enemy order of battle (ORBAT):
    - Land,
    - Air,
    - Navy,
    - Logistics.
- Enemy's current status of deployment and re-arrangement activities,
- Intelligence data for air operations,
- Information on conventional, chemical and nuclear targets,
- Counter intelligence data,
- Information on targets and command control objectives,
- NBC information,
- Enemy's electronic warfare order,
- Enemy's air support for naval operations.

c) Operations

- Military restricted and security zone activities,
- Intrusion of land borders, territorial waters and air space violations,
- Control of air space and activities for utilization of military airfields,
- Harbor visits by foreign military ships and the activities for utilization of harbors,
- Alert status,
- Activities related to war headquarters,
- Organizational activities and activities related to deployment and establishment,
- Doctrines and plans,
- Government/Command Constraints/Criteria,

- Government directives and criteria,
- Friendly forces order of battle,
- Rules of engagement,
- Transfer of authority,
- Conventional, chemical and nuclear targets,
- Sea surface data,
- Lines of communication,
- Sea transportation.

d) Logistics

- Current logistics situation/order,
- Combat vehicle and equipment war reserves,
- Medical supplies and medical evacuation and treatment,
- POL (Petroleum, Oil, Lubricant) storage and distribution capabilities,
- Roads,
- NBC equipment,
- Arm stockpiles,
- Maintenance and repair capabilities,
- Supply material.

e) Plans and Policies

- National strategies,
- Alliance strategies,
- Force planning,
- Defence planning,
- Financial plans and programs,
- Agreements.

f) Communication /Information Systems

- Communications requirements
- Information system requirements,
- Frequency plans and management,
- Communications security,
- Standards,
- Software development and maintenance.

2.3. REQUIREMENTS ANALYSIS AND COMPUTER AIDED SYSTEM ENGINEERING

The analysis of operational requirements is the most important part in effectively carrying out the design of a C^3I system. This analysis can be divided into two general categories as functional analysis and data analysis.

2.3.1 Functional Analysis and Data Flow Diagrams

Functional analysis aims at determining the activities performed in the organization, to what extent these activities are performed and what other new activities will be needed in the future. In other words, it determines the current and future functional mechanisms and the mechanisms required to be planned. The core of functional analysis is functional modelling. A functional model includes functions/processes performed in the organization, the events triggering these activities and distinguishing entities manipulated by these activities and the attributes of such entities. Throughout this modelling, the objectives of the organization are kept in mind. The principal goals of the functional analysis and modelling can be stated as:

- To establish a base and a reliable model for systems which have already been developed or are planned to be developed,
- To put forward a model that is independent of any mechanism or method of operation so that objective decisions will be made while choosing among alternative system designs.

As in all modelling studies, functional analysis is also based on powerful standards and techniques. It may be desirable in some cases to use more than one technique together to be able to cover different aspects of the organization. Each functional modelling technique requires different conditions to be met. In the following subsection, a functional modelling technique, called Data Flow Diagram (DFD) [2.2], which is one of the most commonly used will be described.

DFD is a simple, yet powerful and easily understandable graphical tool. It is based on the principle that functions can not be carried out unless some data is received from another source such as another function, a data store or an external entity. It considers information systems from the point of data movements and models the real world tasks at various levels of detail. Data movements may be real or imaginary. This technique which is very useful for functional analysis phase determines the borders of the tasks to be automated. A well designed DFD describes in detail the organizational structure and the functional requirements of the organization. It is possible to view the whole system or the subsystem at any detail through a corresponding level of the DFD. This property of DFDs establishes a common terminology and a common point of view between the users and analysts for the information system to be designed. The useful features of this modelling technique are:

- It depicts the structure of the C^3I system to be designed completely.
- It allows top-down design without loosing the details.
- It is easy to follow and present.
- It is easy to define interfaces to the external world.
- The outline of the system can be shown to the end user.
- The boundaries of the system and the subsystems can easily be determined.

On the other hand, in cases where the requirements change rapidly, updating of the diagrams becomes laborious and for uncommon applications the designer may have to introduce new constructs not normally used in the model.

The highest level DFD is called a context diagram. A context diagram shows the general system outline, the boundary of the system and the relationships between subsystems. Lower level diagrams are used for:

- Cross-checking the functional hierarchy,
- Depicting functional interdependence and mutual independences,
- Detecting triggering events,
- Providing for cross-checks to detect and control relationships between functions and data.

The basic processes which can not be divided further correspond to the programs and procedures in the software development phase. These processes take the state of the organization from one consistent state to another or do not change the state. Such processes either occur or do not occur at all. If a process visits some consistent states while it is running then this process is not basic and should be further divided. The information that these processes contain are:

- Description of its function,
- The entities, their corresponding attributes and the functions (read, write, delete, etc.,) performed on them,
- Authentications,
- Related documents, rules, and mechanisms,
- Frequency,
- Triggering and triggered events.

A simple dataflow diagram is shown in Figure 2.2.

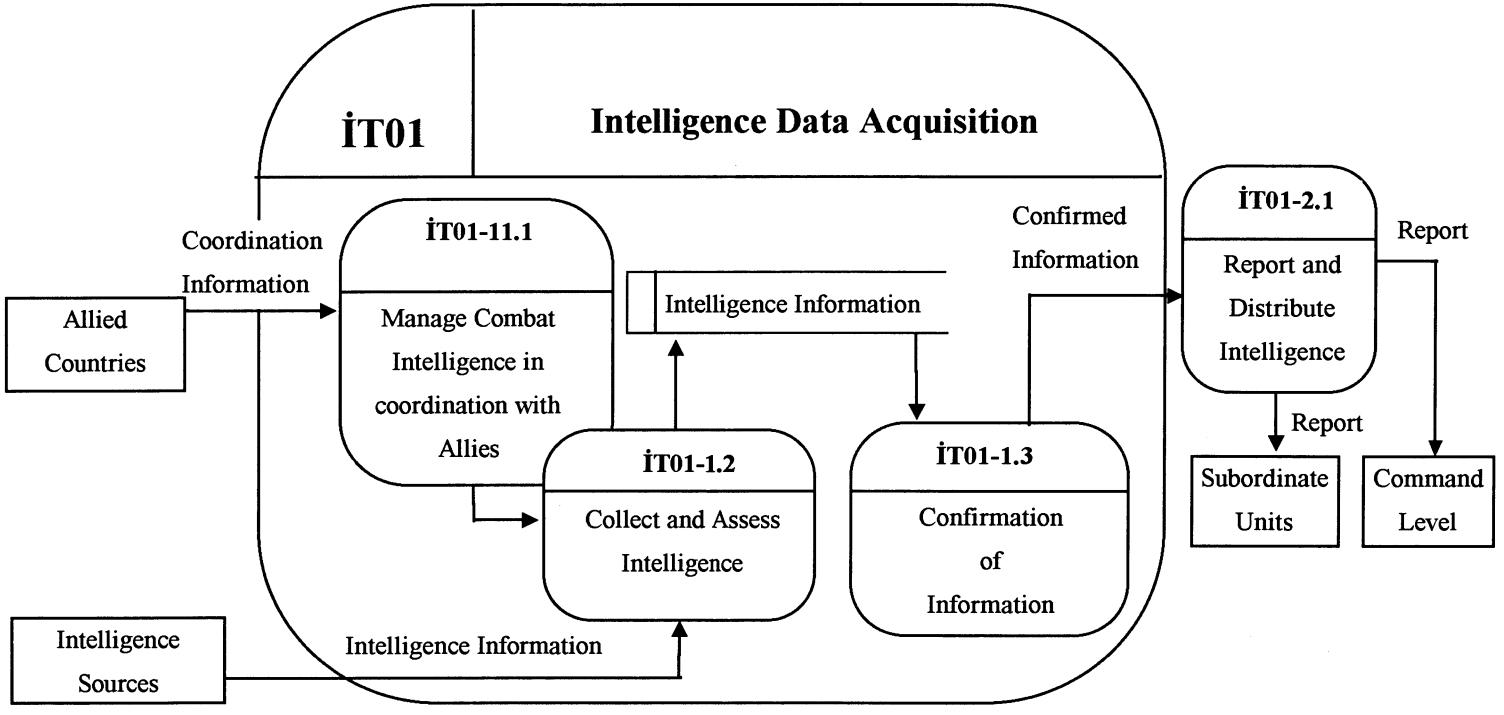


Figure 2.2 A Simple Data Flow Diagram

2.3.2 Data Analysis and Modelling

Data has prime importance for a well organized and effective information system. It is for this reason that, besides functional analysis based on processes, an analysis based on data should also be performed. Data analysis may sometimes gain even higher priority. Management Information System (MIS) divisions of organizations have mostly adopted the system development methods based mostly on data analysis techniques. In the context of the design of a C3I system, therefore, it is clear that not only functional analysis but also data analysis should be conducted in parallel.

Data analysis is based on the principle that the integrity of information systems can only be assured through a common logical model dictated by centralized control. This principle implies not that the information system will have a centralized control or structure, but that the data models will be maintained centrally. Data modelling and analysis asserts that how the information is used, shared and managed can be determined by examining all the information manipulated in the organization. It requires that the program structures should be established on top of the data model. This data model is constructed in regard to the whole of the organization so that it describes overall the underlying structure of data. In summary, programs are designed after the data modelling and analysis phase and the quality, i.e., efficiency and performance, of the information system are directly related to the quality of the data model.

The collection of data of the organization is a lengthy and difficult task. Some powerful techniques have been developed to facilitate this task and to ensure that all data are defined correctly and consistently. Among such techniques are the Entity-Relationship Diagram (ERD) developed for high quality data modelling and analysis as a tool of information engineering [2.2]. Entity Relationship (ER) models allow the definition and description of data and relationships between them to be specified in a standardized way. It is this property that makes the ERD a universally accepted tool as a modelling technique.

There are two kinds of analysis in the context of data analysis:

- **Entity Analysis.** It comprises the determination of entities and relationships between entities which are significant in the operation of the organization. Entity analysis is the most important part in the context of data modelling. This process involves the derivation and determination of the initial, average and maximum sizes and growth rates of these entities and frequency and type of access to these entities. Meanwhile, the format, average and maximum lengths of the attributes of the corresponding entities, and types of the attributes as to being mandatory or optional are decided.

- **Data Analysis.** It is the inspection of entity analysis outputs using known and registered data; hence they are adapted to the normalization rules and documented in a formal language. For this purpose, the reports, files and graphics can be examined. This process is also called bottom-up checking.

2.4. DERIVATION OF SYSTEM DESIGN PARAMETERS AND FEATURES

The design of a C^3I/MIS is considered complete only when the following parameters and aspects of the system are specified based on the operational requirements:

i) Users of the system and how they are organized
ii) System configuration
(links, nodes and redundancy)
iii) Information Exchange Requirements
(source/destination, man-machine interface, type, volume/frequency, databases, speed and security level)
iv) Information Processing Requirements
(Core capability, application software and implementation priorities)
v) System Capacity
(transmission, switching, storage and processing)
vi) Generic charactistics
(Security, survivability, reliability and interoperability)
vii) System Specific Characteristics
(Open System Architecture (OSA), distributed architecture, COTS, MLS, functional integration, evolutionary development and procurement, reliability/survivability and interoperability)
viii) Costing
(Capital and recurring costs)
ix) Cost and Time Reduction Measures (Strategic Goals)
(user productivity, software development productivity, portability/reusability, vendor independence and generally life-cycle cost reduction measures)
x) Constraints
(Financial, time, technical, legal, man-power and political)

How the design parameters/features which may be generic or system specific are derived from the user requirements are outlined below. In this connection it should be pointed out that for best results, continous coordination and cooperation between the user and designer/developer is necessary to ensure consistency, integrity and completeness of the system design. It is also to be noted that the so-called system specific requirements become generic i.e need not be specificly mentioned by the user, as technology advances and the user needs evolve with time. For instance; the requirement for OSA which provides for vendor independence and interoperability; distributed architecture which increases availability and survivability and evolutionaray development and procurement which ensures that the operational system can adapt to changing user requirements and technology, are becoming generic because they can be met economically and provide features which are regarded by most, if not all, users as desirable.

The design parameter/features related to items (i), (ii), (iii), (iv) and (v) above are directly derivable, as shown below, from the data supplied by the user as captured in Table2.2, 2.3 and 2.4. The system configuration (item ii)consisting of nodes and links is mainly determined by the locations of the major users/nodes (item i) and by

the minimum-cost network requirements as well as by survivability considerations (item vi). These issues are dealt with in full measure in Ref [2.3].

It will be appreciated that the design parameters/features affect one another as shown in Fig. 1.3 and the cost of the system depends among other things on the redundancy/hardening and mobility provided in the system for survivability/availability which can be adjusted in an iterative manner consistent with the constraints.

The efficient design of a very complex structure such as a C^3I system requires design tools and the tools that are in question here are the Data Flow Diagram (DFD's) and the Entity-Relationship Diagram (ERD).

The Data Flow Diagrams (DFDs) are obtained by using decisions/actions and the data sets related to these decision/actions for each C2 activity category as depicted in Table 2.2. The level of detail for DFDs may change from a basic process for a decision/action to a complete activity comprising a set of decisions/actions. DFDs help the designers to define the functions performed in the organization and provides a way of checking incomplete and missing requirements. DFDs may be used for prioritizing the application software to be developed (item iv)as explained in Chapter 6.

The processes depicted in the DFDs requiring interaction between users when grouped with respect to functionality give an estimate for the number of servers. Additionally, the decision/action field shown in Table 2.2 when grouped with respect to the type of processing gives us the necessary software packages required for core capabilities such as message processing, video conferencing, database management systems and briefing support. This allows the core COTS products to be specified.

The Entity-Relationship Diagrams (ERDs) are obtained by using the data in Table 2.3 and 2.4. ERDs, when completed, allow the design of database for the various functional areas. The distribution of database files between organizational units, file transfer, access and sharing requirements between nodes, security levels of records in database tables and replication needs are readily obtainable from these diagrams. Approximate initial sizes of the database tables can be estimated by simply multiplying the number of records by the corresponding record lengths for each table. Length of a record is known at this point since the data type and the length of the fields have been determined. Calculation of the number of records for a table, on the other hand, is made using the data available, either automatically or manually, within each related branch of the organization. The number of currently working personnel, for example, may be taken as the number of records in the employee table.

The processing requirements are determined depending on the type of each task to be automated within the organization. A task is computationally defined in terms of the type of data to be processed and the type of processing (e.g. number-crunching, accounting, image processing) which are both extracted from Table 2.3. As a result, the computational complexity of the task and volume of data together with the

response time constraints dictate the specifications such as internal and external storage capacity derived from the volume of data to be processed, instructions executed per second depending on the computational complexity, response time and the type of data to be processed, arithmetic co-processor need to speed up the compute-intensive operations and vector or raster graphics capabilities for graphics applications.

The source, destination, volume and frequency fields of Table 2.4 allow us to estimate both the internodal and intranodal communications traffic (see Annex 6-A). With this information in hand, type and architecture of the LANs and WANs can be determined which, in turn, enable the necessary protocols compatible with the respective LAN and WAN architectures to be specified. Security of communications links is also classified with respect to the highest security classification of the information conveyed in each link which are readily obtained from the data in Table 2.4.

After specifying the system as above, system maintenance and operation aspects including manning, training and logistics requirements are determined as explained in detail in Chapters 8 and 9.

2.5 TESTBEDDING AND PROTOTYPING

A formal and complete requirements analysis methodology as described in this chapter is essential to the proper clarification of design data to the subsequent implementation of a major C^3I sytem. As a supplement, however, it is carrently strongly advocated to perform in parallel significant efforts of user oriented testbedding and prototyping. The main goals of this activity is two-fold.

Fristly it allows for rapid introduction of new technology. The speed by which certain areas of information technology develop today, makes it critically important to find ways to rapidly exploit new technology. This is achieved in combination with evolutionary acquisition by developing simple prototype implementation of relevant functionality and involving the future users through experimental testbeds and field applications during exercises and training operations.

Secondly it allows the users through this exposure better to understand their own needs and especially how these needs can be satisfied in various ways by modern technology. The extremely useful dialogue that develops between system developers and future system users and operators can rectify many misunderstandings that otherwise could develop, with often extremely costly consequences.

To exploit technology in an efficient way competent scientists must be provided with laboratory facilities equipped with modern technology so that through hands-on work they are able to know and understand the capabilities of modern technology. When the developers know and understand modern technology, they must then understand how that technology can be applied in operational systems to accomplish functions which are useful for military purposes. This can be done effectively only through a process of user oriented testbedding and prototyping, since such a process provides an environment in which the real userand the system developer are able to interact in an operational setting when assessing the utility of protential system capabilities.

When the user and system developer are convinced that they identified a useful set operational capabilities, those capabilities must be implemented quickly if technology is to be efficiently exploited. This can only be achieved through evolutionary development and acquisition. A number of examples of recent experiences gained at the SHAPE Technical Centre in this area is provided in [2.4].

The most common approach to system implementation in the C^3I area is through competitive acquisition of technology and capability available in the commercial sector. Scientific experimentation and prototype development should play a key role in executing such a strategy. Prototyping can be used to reduce overall acquisition risk in instances when user requirements are not fully understood, where there are unusually challenging or unproven system elements or when the required system is large and complex. A prototype can be implemented by the system developer and evaluated by the user to better define and understand needed capabilities, especially in the case of information systems. This directly supports the development of acquisition specifications. Prototyping can also be used to demonstrate the viability

of certain key technologies or technical solutions. Prototypes can generally be of two types: Short Term (Disposable) or Longer Term (Evolutionary).

Disposable prototypes fall again into two catagories: research and requirements definition. Research prototypes answer specific scientific questions through experimentation and the end result is typically a technical report. Requirement definition prototypes, on the other hand are intended to aid in understanding operational requirements, and must sustain limited fielding in order to be exposed to its true environment. The use of this type is as stated above to facilitate the dialogue between the user and the system developer "on the users home ground" and the end result is typically a validated user requirement forming the basis for a procurement specification. Disposable prototypes are allowed to be developed "cutting corners" in order to speed up the development time and must be expected to fulfil all the requirements normally required of the final system. However, in today's environment, urgent operational needs frequently cause these prototypes to be fielded as interim operational capabilities, later to be replaced by the final system.

Evolutionary prototypes are intended to refine user requirements or to illustrate not yet fully identified needs. A good example is the frequent incremental use of software application releases in so-called "beta stage". These releases are used to generate user feedback, forming the basis for the subsequent operational releases of the applications. For obvious reasons, the major elements of the evolutionary prototypes must be rigorously documented and must be developed in highly controlled and high quality development and implementation environments.

2.6 REFERENCES

[2.1] Andriole, S. J., Halpin, S.M., "Introduction: Perspectives on Command, Control and Information Technology", Information Technology for Command and Control, IEEE Press, pp. 1-7, 1991.

[2.2] Downs, E., "Structured System Analysis and Design Method Application and Context", Prentice Hall Int., 1992.

[2.3] Ince, A. Nejat et al, "Planning and Architectural Design of Integrated Services Digital Networks", Kluwer Academic Publishers, Boston, 1995.

[2.4] Diedrichsen, Loren D. "User-oriented prototyping: the basis for cost-effective procurement of Military Communications and Information Systems" Proceeding of the 1995 AFCEA Turkiye Seminar: "Digital Revolution for the Military".

CHAPTER 3

GENERAL SYSTEM OUTLINE
goal architecture

3.1. SYSTEM CONCEPT

In designing a complex information system, it is not always possible for the user to be able to specify all his requirements prior to the design. It is, therefore, necessary to follow a goal driven design process. Such a design necessitates to put forward a generic goal architecture based on both the generic and some system specific characteristics together with some strategic goals. These constitute the "system concept". Drawing a framework for the system concept and the use of a Reference Model (RM) facilitate the design process and minimize the risk of missing user requirements. Such a goal driven design process enables also a meaningful and effective transition from the implementation strategy to fielded systems which would otherwise be non-realizable for the whole system with the current state-of-the-art technology.

The basic functional requirement of a C^3I system is that each user should have access to the necessary information and the adequate data processing support necessary to perform his duties, regardless of the physical locations of the user and these resources. A conceptual model could then be given as shown in Figure 3.1.

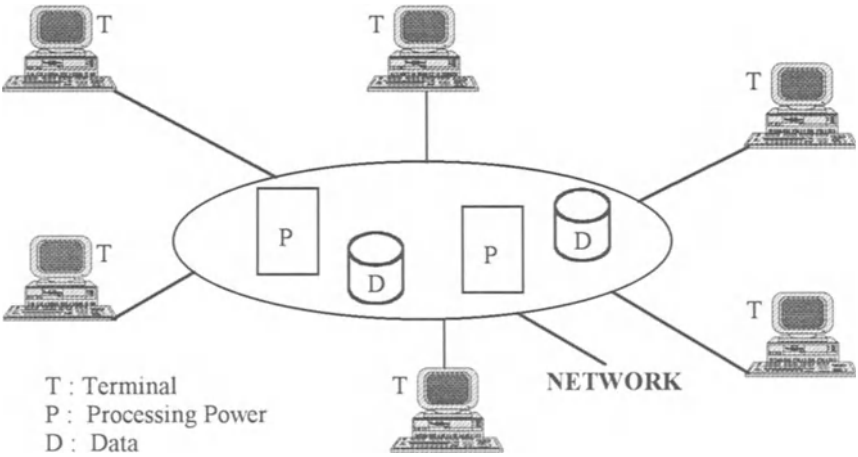


Figure 3.1 A Conceptual Model

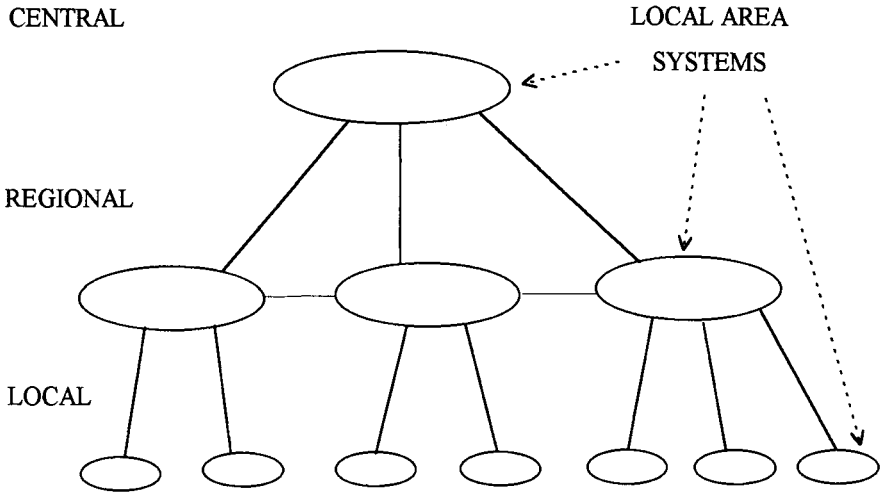


Figure 3.2 System Level Model

For the organizational structure given in Figures 1.1 and 1.2, of Chapter 1 the conceptual model can be rationalized as shown in Figure 3.2 where the lines represent not physical but logical connections.

Figure 3.2 shows an architecture that consists of interconnected local systems serving departments/units at various levels of the organization. Within each of these local systems, there exists a network. In this network powerful resources, i.e. central computers, can be freely combined with departmental computers and various personal resources such as workstations and inexpensive PCs and terminals. Each node at any level, central, regional or local, can access resources on any hierarchy level as long as the access controls allow. The generic characteristics making such a goal architecture possible are explained in the next section.

3.1.1 Generic Characteristics

Generic characteristics which are those that every system must have upto a degree, and which are part of the system concept are determined with respect to a set of criteria dictated both by user requirements and by the need for smooth operation of the system. The generic characteristics of the C^3I system to be designed, are outlined below.

3.1.1.1 Survivability

Survivability is a feature demanded by both civilian and military systems. Survivability is the ability to resist external influences and for certain predefined parts of a system to operate autonomously when cut off from the rest of the system. For a detailed study and design of survivability reference to [3.1] should be made.

Survivability can be achieved both physically and electronically. An important physical aspect of survivability is autonomy. The degree of autonomy may change from the actual isolation of a unit from the rest of the system to re-establishment of connectivity.

Another, but a more expensive solution may be mirroring. Crucial systems and services can be duplicated at the same or different units of the organization. In normal cases these dual systems can share the load of service or one of them can be reserved only as a mirror, while the other is serving its main tasks. However, one should not disregard the fact that keeping integrity of replaceable systems may result in unbearable overhead. Of course the system can be designed so as to promote the combination of the two approaches to achieve the feasible and not necessarily the most survivable system.

3.1.1.2 Security

Security requirement is a complicating factor. Complying with security requirements on a network wide basis is highly dependent on operational and technical concepts. Information exchange by means of formal messages is a typical example of a concept that is relatively straightforward to handle and is very attractive with respect to security as well. Remote log-in is an example of a concept which is straightforward from a technical point of view, but raises a number of questions with respect to security, related to access control.

3.1.1.3 Reliability

Reliability is yet another operational requirement having impact on the technical configuration: High reliability usually implies reliable components and higher complexity as well. There is a distinction between survivability and reliability. Reliability is the ability to operate, under given conditions (temperature, humidity, vibration, etc.) and is typically defined by such figures as MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair).

3.1.1.4 Interoperability

The goal architecture which will serve as a basis for implementing operational as well as administrative systems, must allow for local autonomy and for both horizontal and vertical interoperability. The complexity of connecting local systems into a completely integrated system is highly dependent on the level of ambition with respect to integration. For example, for information exchange, one extreme could be the exchange of flat files, the other extreme applying a distributed database concept. Commonality of operational procedures, data elements, data model messages, data dictionary and forms library are cornerstones in achieving interoperability.

3.1.2 System Specific Characteristics

The characteristics of the system derived from functional area specific user requirements, are the ones shaping most applications and dictating most of the design process. Any serious information system designer and implementor will recognize the importance of listening to and cooperating with the end users. The result will in many cases be solutions tailored to the single user, user group or organization, i.e., not something widely applicable throughout the organization or between organizations. This user recognition of own needs in terms of data processing support will evolve continually as a function of the evolving organization and the user's perception of what he can and should expect from the market as well. The architecture applied should accommodate the need for continuous change. At this point the importance of user driven development becomes obvious.

3.1.3 Strategic Goals

In designing a C^3I system, the feasibility of implementation should always be taken into consideration. Minimization of system cost, ease of operations and maintenance, training facilities, scalability of the system are examples of strategic goals.

3.1.4 Common Technical Solutions

To satisfy generic and system specific characteristics and strategic goals, common technical solutions are required wherever possible. Common functional specifications, common applications, common communication profiles and network services which can be based on common development tools, common operating system and basic executive software and common hardware are key factors in achieving a successful design meeting C^3I system requirements.

Common solutions contribute to the reduced cost of development and acquisition, operations and maintenance, and training. Incremental growth is another strong cause for commonality: It is very hard to imagine how it is possible to make system upgrades without paying attention to the existing configuration.

Turning to generic characteristics and their influence on commonality, a natural consequence of a common information base and common operational procedures are common functional specifications and consequently common applications. Common communication profiles and network services are naturally derived from autonomy and interoperability requirements.

Fulfilling the requirement of commonality is possible by adopting international standards. An international standard with a high degree of acceptance in the commercial market is definitely the ideal case. When international standards or treaty-based regional Standards [3.5] are not available, voluntary de-facto widely recognized commercial standards are preferable.

With all these in mind, goal architecture and basic information processing functions for the C^3I system will be explained in the next section.

3.2. BASIC INFORMATION PROCESSING FUNCTIONS

The layered approach, that is outlined via the Reference Model will be followed in the design of the overall system. In this section a general goal architecture will be presented and the principle information processing functions of the C^3I system will be described. We can start with a generic configuration for the C^3I system which is dictated by the user requirements (Figure 3.3).

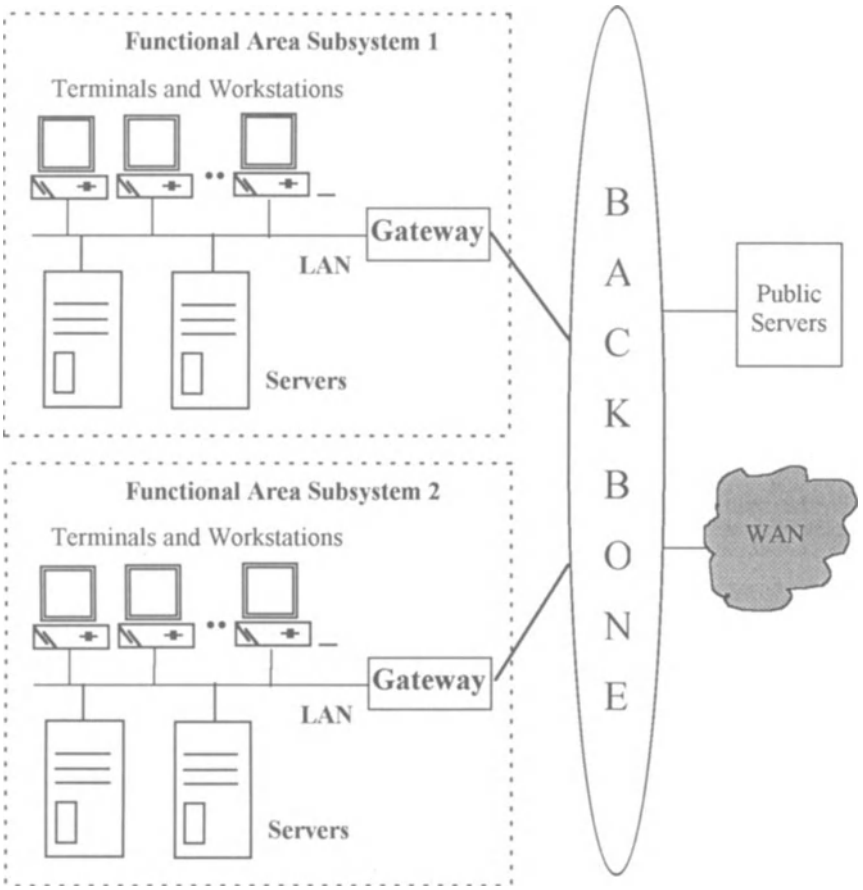


Figure 3.3 Generic Configuration of C^3I System

It is important to note that; the generic architecture, depicted in Figure 3.3, is a rudimentary view of the overall system, rather than an imperative design schema. The term FAS [3.2] (Functional Area Subsystem) refers to the autonomous units performing specific tasks and functions and can be of any size (a unit, branch or a

department). Correspondence between organizational structure and FASs mainly stems from functionality and geographic dispersion.

FAS's are set up on their private LAN's with sufficient computing, storage power and I/O facilities and they are interconnected via an organizational backbone. Services common to all FAS's, can be provided through public servers which are connected to the backbone. The backbone can be designed in any topology and size (although shown as ring in the figure) to meet the requirements. Gateways in FAS's serve as access points to the backbone and manage routing and access control.

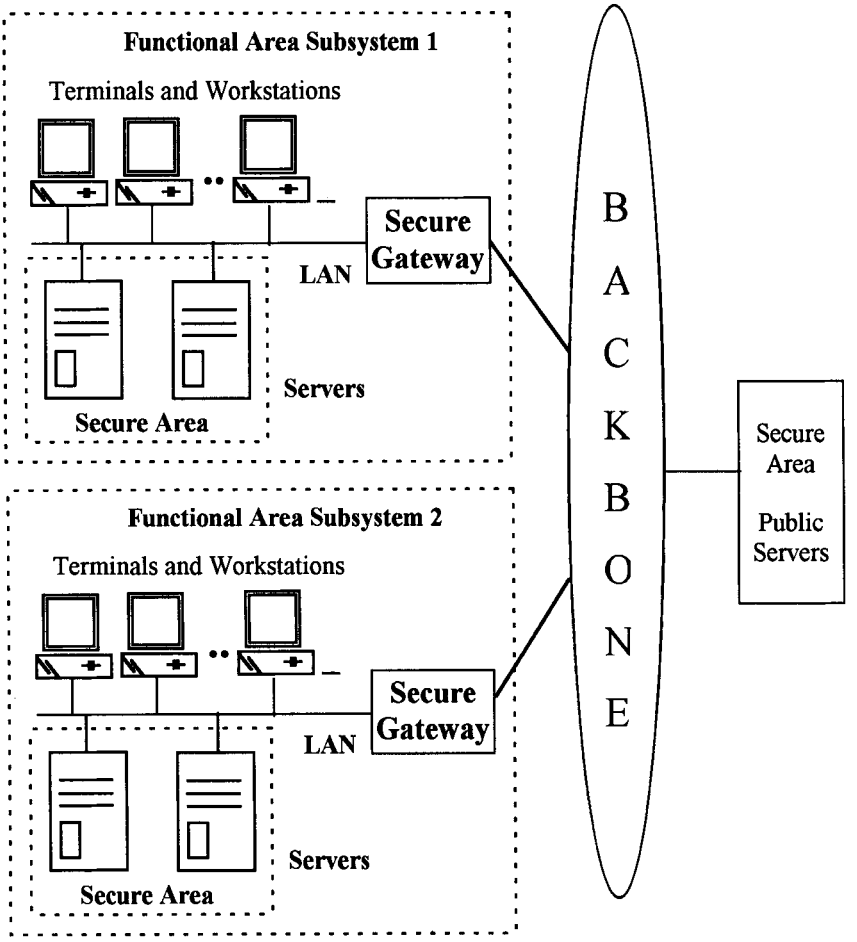


Figure 3.4 General Architecture Meeting Security Requirements

When security requirements are taken into consideration; the generic architecture is modified. For example servers providing information service (information service will be explained in 3.3) and being publicly accessable may be undesirable. The configuration shown in Figure 3.4 carries some of the common servers onto the LAN's which are planned to be set up in a secure area. In this case the gateways

connecting FAS LAN's to backbone are sophisticated gateways performing security functions. As a consequence all the servers, operating systems and applications are expected to meet the desired security requirements.

3.2.1 Man Machine Interface

Today the number of computer users are dramatically increasing and computer based culture in the user communities is developing over a wide spectrum, while emerging technologies both in hardware and software are getting more complex. Designing software systems requiring minimum user intervention in performing tasks without loss of feeling of control, is an important area of concern. Man machine interface is an information channel that conveys information between the user and the computer [3.3]. This channel defines the rules and protocols constituting the structure of the interaction, interpretation of user actions by computer and how variously formatted data are presented to the user.

Several criteria can be put forward to measure the quality of the user interface. Effective systems generate positive feelings of success, competence, mastery and clarity in its user community. The users are not restricted by the computer and can predict what will happen in response to each of their actions. When an interactive system is well designed, the interface directly disappears, enabling users to concentrate on their own work, exploration and pleasure. Creating an environment in which tasks are initiated almost effortlessly requires a great deal of hard work from the designer [3.4].

Starting with main goals that user interfaces are expected to accomplish, preliminary design criteria can be more easily determined. The U.S. Military Standard for Human Engineering Design Criteria [3.4] states the purposes as :

- To achieve required performance by operator, control, and maintenance personnel
- To minimize skill and personnel requirements and training time
- To achieve required reliability of personnel-equipment combinations
- To foster design standardization within and among systems

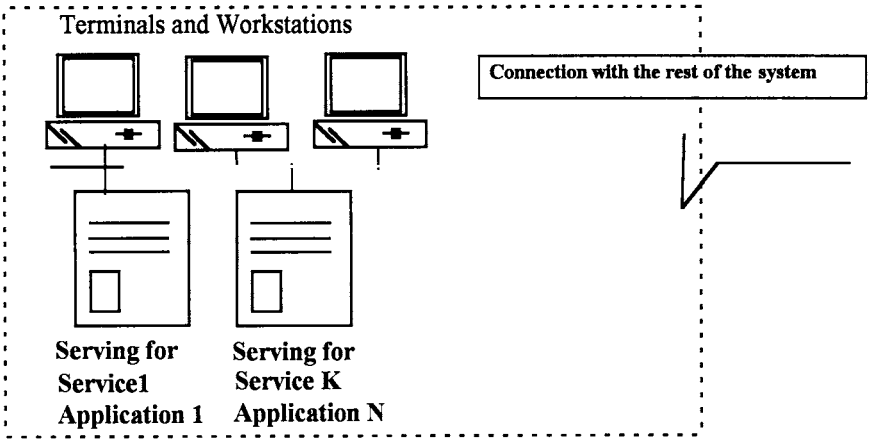
More specific goals can be determined by analyzing the requirements of diverse units and user types. Proper attention to factors related to diversities in user education levels, work loads, computational skills and rigorous testing often leads to reduced cost and rapid development. A careful tested design generates fewer changes during implementation and avoids costly updates after release of new systems.

In C^3I systems the purpose of the user interface design is to provide the user, with a consistent and integrated application environment which hides the overall infrastructure of the system. These requirements are explained below.

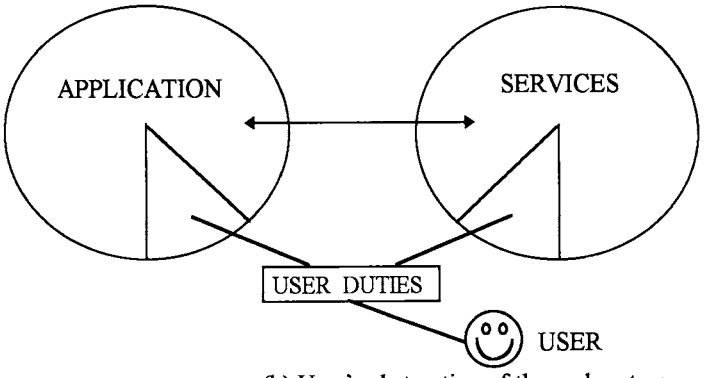
Consistency: Consistency dictates that, the appearance, expected user actions and application behavior should be similar and predictable. Navigation in and among applications should not cause awkward changes in the environment.

Integration of applications: The user or functional area specific applications should be integrated with general purpose system and network services and applications such as e-mail, file transfer services allowing identical "look and feel" when moving from general to special applications.

Infrastructure hiding: The user and the organization should not suffer from the underlying infrastructure of the overall system. There may exist hundreds of computers giving lots of different services and serving applications in C^3I systems. Underlying procedures and protocols executing primitive tasks must be hidden and a high level system view should be exposed to the user without lack of service availability (Figure 3.5).



(a) Real System Designer's and Administrator's View



(b) User's abstraction of the real system

Figure 3.5 Infrastructure Hiding

3.2.2 Data Management

C^3I systems demand determination of the most efficient and feasible policy of data management. Similar to furnishing a house and keeping it tidy, the problem of data management consists of processes of the following tasks for a C^3I system:

- determining a strategy for partitioning and positioning of the data to the sites in the most efficient manner,
- giving well defined responsibilities and correct access rights to the sites
- ensuring consistency and integrity between partitions
- minimizing the risk of database failures arising from different sources; that is achieving suvivability of data
- Recovering errors and lost data due to system failures

Sophisticated models and methods for data organization and access have existed for about thirty years as a result some standards such as database languages, distributed and client-server protocols are quite mature. Consequently, some of the tasks listed above does not require additional development. Several vendors provide database management systems allowing distributed and client-server applications, however because of the proprietary standards interoperability between these vendors is not readily acquired in heterogeneous environments. Achieving interoperability between such multi-vendor nodes requires overwhelming integration effort. Bottom-up design approaches based upon integration of currently used databases, are candidate cost-factors for increased system cost.

3.2.3 Applications and Services

The total set of interrelated tasks executed to meet organizational goals are generally grouped into subsets based on the degree of interrelationship between them. Closely related tasks are grouped into subsets and the responsibility of executing each subset is assigned to a department or workgroup forming functional areas. There will be several functional areas in an organization depending on the size and variety of tasks that need to be executed. As a result the applications and services used in the organization may be different at each functional area but these applications may generally be divided into two major groups: common and special applications [3.2].

3.2.3.1 Common Applications

Common applications (or services) are the type of services which are required by several functional areas; generally by almost all functional areas. Message Handling , file and database services, word processors are examples of common applications. Common applications have three main aspects [3.2]:

- Reduce application development effort by offering ready-to-use common services. By providing readily a set of frequently required services, the application development effort may be focused on functionality rather than providing the underlying technical infrastructure. This will reduce the development effort and hence the development time.

- Ensure application interoperability by offering standard solutions to common requirements. By offering standard solutions to frequently required services, multiple and incompatible solutions to the same problem may be avoided.

- Other standard interfaces between the computing platform and applications. Application interoperability and integration of different applications most notably require standard set of services. Solutions based on proprietary systems and standards will cause interoperability problems.

Since it is commercially available, software in the class of common applications constitute a wide spectrum of alternatives. As a result, a selection among the possible candidates for a common application requires a compromise. Important aspects of this choice are as follows:

- User preference may change and applications utilized throughout the organization may vary from site to site or even from room to room. This is undesirable from management and administrative point of view so it is very important to keep common applications interoperable.

- Common applications must support standard network services.

- Common applications must allow central configuration but also user based customization.

- Since expected to be integrated with other applications in the C^3I system, it is also important to choose applications that are compliant with the user interface environment chosen.

- Applications must comply with security standards at the desired level.

3.2.3.2 Special Applications

Special applications stand for functional area or personnel specific tasks. The applications of this type are seldomly available commercially and they generally require in-house or a third party development. Since special applications are also be used by a community of users the same criteria for common applications applies for both procurement and development of them. Besides, modern software engineering paradigm dictates the following specifications to be acquired, to ensure the software quality.

Scalability: The software or its modules may be scaled linearly to accommodate new users and evolving requirements.

Reusability: The software or its modules can be used in applications with different purposes.

Portability: The software must support different hardware and software platforms without lack of service.

3.2.4 Information Exchange

A C^3I system can be viewed as a network of pipelines in which information flows bi-directionally among nodes. To determine the structure of information flow is very important to keep the information up-to-date, increase the confidence and minimize the filtering of information. Information exchange requirements can be analyzed in three groups.

Automatic: The information exchange is handled by a distributed database management system through replication facility and once the data is replicated, the management of them is transparent to the user. Data updates are transferred to pertinent sites automatically. Some special applications, such as message handling systems, can be tasked to manage automatic database updates. This is the most desirable method since the exchange is fully under the control of the designers and system managers and security can be more easily achieved.

Semi-automatic: Users exchange information via network services such as file transfer, remote logins and remote procedure calls and e-mail. Protocols and data exchange formats (document, image, voice, video formats) must be predetermined and must be kept uniform throughout the organization.

Manual: Refers to the exchange of archiving devices such as diskettes and magnetic tapes.

One of the main functional requirements of any C^3I system is its capability to provide the right information, at the right time, to the right person or application, in the right form to be assimilated and acted upon or to be processed, especially in time critical situations. The user in the system must be able to trust and be confident that the information is available and is indeed the right information. The types of information exchanged within C^3I system will be text, data or image.

The C^3I Communication Subsystem will provide computer networking and communications services that include data communications, message handling services, transparent file access, user workstation support, distributed computing, and network security capabilities. All computer networking equipment and communications systems, and services offered should be specified in accordance with ISO OSI model and related CCITT/ISO standards and all applicable layers referred to profiles, should be explicitly specified.

Depending on the communicating entities five types of Information Exchange (IE) can be distinguished within a C^3I System. These are:

- IE Between a User and Node Subsystem or a Node System
- IE Within a Node Subsystem or Node System
- IE Between a Node Subsystem and a Node System
- IE Between a Node System and a External Site
- IE Between Node Systems.

While the man-machine interface provides IE between the user and the system, the other types are supported by the C^3I system's Local Area and Wide Area (LAN/WAN) network services. The following network services must be supported:

- Messaging (e-mail, X.400)
- File Transfer (FTP or FTAM)
- Remote Terminal (e.g. Telnet)

The security functions provided to secure these exchanges of information must contain the appropriate combination of labelling, need-to-know seperation, access control to meet the requirements for protection against defined threats.

3.2.5 Interoperability Services

The C^3I System of interest to us is a distributed, open system covering all the sites or units of an organisation. Each unit is responsible for a set of well-defined, clearly related and interlinked C2 tasks. These closely related tasks define a functional area, and chained tasks define a succession of command or management. Both set of tasks require exchange of information and transfer of commands and management orders. Members of a functional area and command or management chain will:

- communicate frequently with each other and, to lesser extent, with members of other functional areas or command or management,
- share specific data for the functional area within command or management chain, and
- sometimes, need specific data from other functional areas or command or management chains.

Types of exchanged information can be formatted and unformatted messages, bulk data, document, graphics, geographical data.

A number of standard data transfer services will be used for information exchange within a C^3I node and between nodes. This will necessitate the design and development of the C^3I system as a network of interoperable nodal systems.

A C^3I system should be a system of interoperable node systems, designed and implemented from a common architecture including a common set of standards to provide the same interoperability services for the exchange of information.

The interoperability between various system elements within a system over that system's lifetime is especially important in the presence of evolving information technology, where considerable advances in the hardware occur every 3 to 4 years, and in software every decade or so. The evolutionary procurement becomes then even more important for interoperability of the system.

The essence of interoperable system design is to identify fundamental features of a system, or its underlying processes, and then rigidly determine the parameters, whilst retaining the spectrum of capabilities needed to encompass the many implementation strategies and to cater for changes in functional requirements and developments in technology.

Within the framework of communications, interoperability is the capability of systems or system elements to communicate with one another and to exchange and use information including content, format, and semantics.

3.2.6 Standardization

The level of international development efforts and investment in commercial standardisation, and the development of related products, software and subsystems are on a scale not experienced in military procurements. This scale has important military benefits of design integrity and development vialability (through large scale user feedback) and lower costs (through large scale production). NATO, for example, has adopted a policy of using commercial standards for military communications through a Reference Model for Open Systems Interconnection and an Open System Interconnection Profile (OSIP) [3.6].

An OSIP framework is based primarily on ISO standards and European Telecommunications Standards Institute (ETSI) Recommendations and perceived profiles. Standards and profiles may need to be enhanced with military features to fulfil requirements such as precedence and pre-emption, security, multi-homing, multi-end point connections.

Above mentioned NATO strategy shall be followed in design, development and implementation of the C^3I system described in this book. However, there may be some instances of deviations from NATO standards for reasons like unavailability of proven software and hardware products. In these cases, other international or de-facto standards will be recommended.

3.2.7 Computer Security (COMPUSEC)

Computer security is the protection of the information stored in a computer system. It focuses on operating system and database management system features that control who can access a system and the data stored in it. Passwords, auditing, backups, security policies, discretionary access control (DAC), mandatory access control (MAC) policies are examples for methods of computer security control

For a computer system there are four methods to provide security protection:

- System Access Control:
- Data Access Control
- Security Administration
- System Design

System Access Control: The method of controlling access to a system is a two-step process which is called identification and authentication. Identification is the way you tell the system who you are. Authentication is the way you prove to the system that you are who you say you are.

Data Access Control: Another important way of computer security is the control of data access. In a shared system there may be a number of users having separate files. Owners of files want to protect their data against either accidental or malicious threats. There are two kinds of controls for data access:

1. Discretionary Access Control
2. Mandatory Access Control

In discretionary access control (DAC) owners decide how to protect their data, whereas in mandatory access control (MAC) the system has the responsibility. In MAC systems everything, that is, users and files have labels.

Discretionary Access Control:
Discretionary Access Control (DAC) restricts access to files (and other system objects such as directories, and devices) based on the identity of users and/or the groups to which they belong. DAC is applied at the owners' discretion, the owner of system objects can choose how to share them. As you tell the system who can access your data, you can also specify the type of access permitted, such as reading, writing, or executing.

Mandatory Access Control
Mandatory Access Control (MAC) is more appropriate for C^3I Systems having sensitive data (government, military or sensitive corporate data). MAC assigns labels to each subject (user, program) and to each object (files, directories, devices, etc.) in the system. A user's label indicates the level of trust associated with that user, it is usually called clearance. A file's label indicates the level of trust needed to access that file. MAC uses the labels of subjects and objects to decide what subject can access what information in the system.

Security Administration: Performing the off-line procedures that make or break a secure system by clearly delineating system administrator responsibilities, by training users appropriately and by monitoring users to make sure that security policies are observed. This category involves more

global security management; for example, figuring out what security threats face your system and what it will cost to protect against them.

System Design: Taking advantage of basic hardware and software security characteristics; for example, using a system architecture that is able to segment memory, thus isolating privileged processes from non-privileged processes.

3.3. COMMUNICATION SUBYSTEM

3.3.1 Description

A fast, secure, and a reliable communication subsystem elaborated with network services and applications is essential for C^3I systems. The design of the communication subsystem is directly related to the organizational structure; since required services and network capacity may differ from site to site depending on the proximity of sites, applications used and network traffic generated. The communication subsystem design includes the determination of:

- Topology of network: The first step in the establishment of the communication subsystem consists of the determination of a relevant network topology. Several factors such as geographic dispersion, cost efficiency, speed and survivability requirements affect the design of the network topology.

- Capacity of connections: Careful analysis must be carried out for traffic estimation to determine the required capacity of connections.

- Standards: Although numerous proprietary standards exist in communication technology there is a trend towards using international standards. It is very crucial to follow international standards in order to evolve the communication subsystem and adopt evolving technologies.

- Specification of network devices: The network devices (routers, bridges, repeaters, etc.) must fully comply with the standards and security requirements.

- Routing methodology: For a given topology, an efficient routing system must be established including alternative routes to achieve reliability and survivability.

- Network management: The configuration must allow network management facilities including remote control of devices, network auditing, and traffic monitoring.

- Network growth forecasts and implementation plans: The inescapable increase in requirements and evolving technologies result in network growth. The

investments must be planned to answer requirements in the future so that it must handle new connections and unexpected traffic increases.

3.3.2 Communication Services and Applications:

The most important application functions requiring communication services are listed below:

File Services: The ability of using a physically distant server's storage unit (or a part of it) as if it is local. This service allows the distribution of file service load on a local area network (Example: Network File Services).

Naming Services: Central naming management of a network is very cumbersome and may cause errors. Naming responsibilities are distributed to local administrators on a hierarchical basis. Name databases are maintained locally and queried by other parts of the network whenever needed (Example: Domain Name Services).

Information Services: Information services try to centralize the local area network configuration database. The need for reflecting even minor changes in the network configuration (e.g. adding a new disk or defining a new user account) on each of the nodes of the network is obviously undesirable. Information services supports a central control and announcing system for such purposes (Example: Network Information Services).

Auditing Services: Monitoring the significant events (e.g. user logins, dial-up calls) on the network is essential. This service is used by network administrators for security and logging purposes.

Network Management: Monitoring the network activity and traffic and remote control of network device and components can be performed by network management services.

Remote Service Calls: This service is used for several purposes i.e. :

- To call services which are not supported locally from other nodes
- Distribute computing power and inter-process communication
- To execute remote applications

Directory Services: Handles the queries about user profiles on a network (Example OSI X.500)

Remote Login: Supports logging in to remote nodes on the network (Example Telnet, Rlogin)

File Transfer: Used in sending/receiving to/from remote nodes. (Example: FTAM, FTP)

E-mail: Provides informal message exchange between the users on the network.

WEB Browsers: Hypertext browsers for navigation through the network nodes giving information services.

Network API's: Including libraries in several languages to access network with primitive service calls. These libraries are used for specific network application development.

3.3.3 Communication Security (COMSEC)

Communications security protects information while it is being transmitted over the network.

Secrecy or Confidentiality. Secure communication keeps information from being transmitted to anyone not authorized to receive it.

Accuracy or Integrity. Secure communication keeps information from being lost, changed, or repeated during transmission.

Authenticity. Availability is a particularly important concept for networks, where even a minor slowdown in service can have a reverberating effect on an entire network.

The communications network further provides support to the security of the system by containing functions that protect the user against denial of service attacks.

3.4 REFERENCES

[3.1] Ince, A. N., et al, " Planning and Architectural Design of ISDN", Kluwer Academic Publishers, Boston, 1995.

[3.2] "AFCEA/Turkey C^3I Methodologies, Modelling and Program Management", Ankara, 1993.

[3.3] Thimbley, H., "User Interface Design", ACU Press, Addison-Wesley Publishing Co., 1991.

[3.4] Shneiderman, B., "Designing the User Interface", Addison-Wesley Publishing Co., 1993.

[3.5] Ince, A. N.,et al, "Digital Speech Processing", Kluwer Academic Publishers, Boston, 1992

[3.6] Jain, B. N., Agrawala, A. K., "Open Systems Interconnection", Mc Graw Hill Inc., 1993.

CHAPTER 4

REFERENCE MODEL

4.1. PURPOSE AND OBJECTIVES

The development and fielding of any new high-tech system requires a very close dialogue and inter-play between the user community and industry.

The reference model (RM) is a functional model and identifies the desired system characteristics so that the concepts of the C^3I system may be conveyed in a standard, non-ambiguous way to the parties, including industry, involved in the system development.

The RM does not identify specific standards or products which are dealt with in the second (Architecture) and the third step (Configuration) in the three-step design process as explained in Section 1.5.

4.2. FUNCTIONAL ELEMENTS

Functional elements of a common reference model are shown in Figure 4.1 and are expanded in the sections below [4.1].

4.2.1 Hardware Platform

The choice of HW platforms is mainly affected by standardization of the Operating System (OS) layer, by the requirements for security and electronic protection (e.g. Tempest).

To fulfill the scalability goal, the architecture must allow HW choices ranging from single user platforms to platforms that can support entire user sites or organization. Furthermore, HW platforms should be scaleable proportionally to changing processing requirements within the individual Head quarters (HQs).

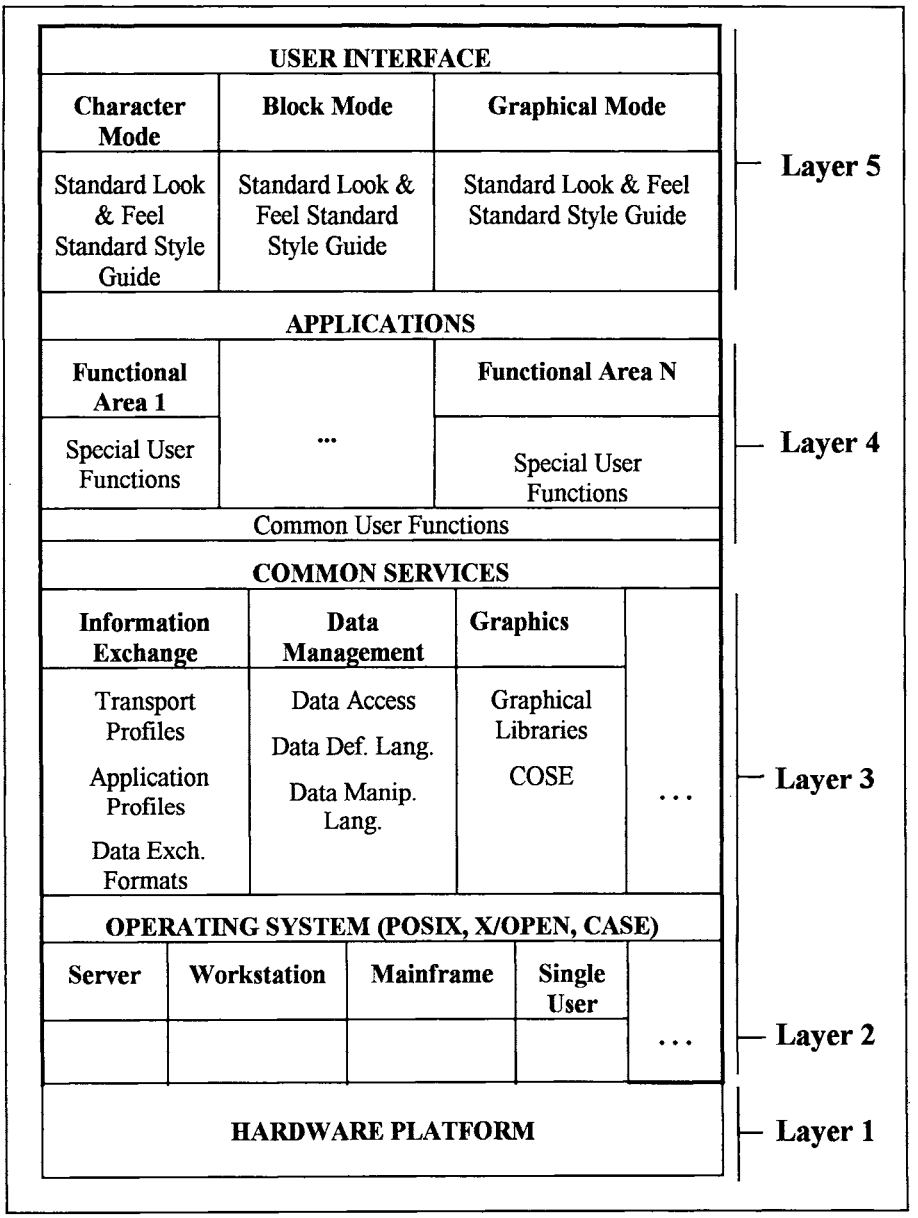


Figure 4.1 A Layered RM with Five Distinct Layers

4.2.2 Operating Systems

Layer 2 defines the Operating System (OS) services required. The OS services provide an interface between the hardware platform and the layers above.

Selecting the operating system is one of the most fundamental choices made in defining standards for the entire architecture. The chosen operating system(s) have direct impact on which standards to select for user interface, hardware, interoperability services and is also one of the fundamentals for obtaining portability/re-usability. Most notably, the choice of OS have an impact on the level of security that may be provided.

Standardizing on one OS within the individual HQs is seen as highly favorable and should be pursued to the maximum extent. A practical standardization could be to establish a backbone infrastructure based on one OS throughout the headquarters or organization which will ensure interoperabilty and resource sharing (e.g., data, terminals, peripheral equipment, etc.). Such a standardization does not negate the goal of vendor independence, and will not exclude the ability to connect special purpose systems based on other OS definitions to the backbone system.

4.2.3 Common Services

Layer 3 defines a set of common services, "common" in the sense that they are required by one or more applications at the above layer. The common services layer has three main purposes:

- Reduce application development effort by offering ready-to-use commonly required services,
- Ensure application interoperability by offering standard solutions to commonly required services between applications,
- Offer standard interface between the computing platform and the applications.

By establishing a set of frequently required services, the application development effort may be focused on providing functionality rather than providing the underlying technical infrastructure. This will reduce the development effort, and hence the development time.

By offering standard solutions to frequently required services, multiple (and often incompatible) solutions to the same problem may be avoided. Interoperability is too often hampered by inherent differences in how the applications interface to Common Services (e.g., different database access method).

Application software interoperability and integration of different applications most notably require a standard set of services. Solutions based on uni- or bi-lateral agreements lead to proprietary systems that may cause interoperability problems.

For portability/re-use of an application software it is essential that applications are interconnected in a standard fashion, allowing them to be removed from one system context and inserted into another without having to also port major parts of their supporting environment (i.e. Common Services). It is thus a prerequisite that the Common Services have well defined and standardized interfaces to the Application Layer eliminating "specialties". For every Common Service there must exist a

standard Application Program Interface (API) that offers the service in a standard manner to the Application Layer.

The Common Services Layer, together with the OS Layer and the User Interface Layer, constitute the ADP infrastructure used to achieve the objectives stated earlier.

Common Services are Information Exchange Services, Data Management Services and Graphics Services.

To fully specify an information exchange service in accordance with the OSI model, services from all applicable layers must be specified. Such specifications are referred to as profiles.

Typical applications profiles that are expected to be available as information exchange services are file transfer, message transfer, remote login, remote application execution, VDU conversation, distributed file system, security.

Data management services are data access and data security. The main purpose of the Data Access service is to define standard access methods towards the various types of data stores, and that these access methods be shared by all applications.

For access to data stored in a relational database the access protocol shall allow all Data Definition Language (DDL) statements and all Data Manipulation Language (DML) statements to be passed to the Relational Data Base Management System (RDBMS), and data returned from the RDBMS to the application. For access to data stored in the file system there must exist one access protocol. In case of systems including different types of file systems, the access protocol shall mask out the differences. Remote databases may be accessed via remote database access protocols.

Database Replication uses a distributed database system to duplicate parts of a local database to a remote site. The size and contents of the segments being replicated depend on operational requirements (functionality, reliability, survivability, performance, etc.). Database Federation on the other hand, uses a distributed database system to join (or federate) physically separated database segments into a virtually seamless global database spanning all units.

Data security shall be provided by a standard set of security mechanisms allowing differentiated access to data based on user privileges.

The Graphics Services provide functions for creating and manipulating graphic objects independent of output devices and to manage the database structures containing the graphics data.

4.2.4 Applications

The Application Layer provide the actual user functionality as specified by functional requirements. This functionality aims at providing automated support for one or more Functional Areas. The applications required to realize the desired functionality is not addressed by the RM. RM is only defining how applications shall interface to the other layers of the model.

From an architectural point of view the main purpose of the applications layer is to:

- provide an integrated application environment
- isolate the applications from the other layers of the RM in order to facilitate portability, component interchange and reuse.
- allow insertion of new applications (i.e. functionality) in existing systems without disrupting existing operations.

Applications shall exclusively utilize the APIs offered by the Common Services Layer and the User Interface Layer. To a certain extent applications may also utilize services offered by the OS, but in general these shall be provided by the Common Services Layer.

Application integration shall not be performed on individual basis directly between applications at the Application Layer, but through the appropriate standard services offered by the Common Services Layer.

Data exchange between applications shall also be handled by the common services. Prohibiting applications to "agree" on non-standard integration methods, ensure that applications maintain autonomy and may be portable/reusable on an individual basis rather than requiring entire systems to be ported/reused. Further, this will enhance the possibility to insert new applications into an existing system and still maintain current operations and the integrated applications environment.

The fundamental concept of the layered RM is to leave the monolithic approach where applications embed both functionality and common services. This monolithic development leads to application that are not easily portable or well suited for reuse in other systems.

Security at the Applications Layer is achieved by using the security mechanisms offered by the Common Services Layer, the OS Layer or the UI Layer. This is ensured by isolating the applications from these layers by standard APIs prohibiting security mechanisms to be bypassed. This also allows insertion of non-trusted applications into a trusted environment if they conform to the model. Development of trusted applications is usually limited to applications that shall "violate" the security mechanisms in a controlled manner (e.g. downgrading).

4.2.5 User Interface

The purpose of the User Interface (UI) Layer is to:

- provide a consistent user interface ("look and feel").
- combine all applications into what the users perceive as a functionally integrated system
- allow resource sharing by allowing access to the data processing resources from one desktop terminal.

National Institute of Standards and Technology (NIST) has developed a User Interface System Reference Model (UISRM) which is a representation of a window system defined in terms of interacting layers as shown in Table 4.1. The UISRM supports client-server operations between clients requesting a UI service and servers carrying out the requests.

Layers 3, 4, and 5 provide the "look and feel" for the UI and at the same time defines the API between application and UI.

The bottom three layers provide the ability to combine applications into an integrated environment. The three bottom layers may also be regarded to belong in the Common Services Layer, but are kept together in the UI Layer to avoid a fragmentation.

To ensure resource sharing and integrated applications, standardization of layers 0 through 2 is imperative. It is of major importance to be able to integrate various systems into what the user perceives as an integrated environment.

Standardization of the upper layers is necessary to achieve a "common user interface" and to allow application portability between different environments and at the same time adopt the "look and feel" of the new environment.

Standardization at the User Interface Layer is to a large extent a direct consequence of the selected operating system standard (e.g. UNIX based OS leads to X windows, DOS leads to MS-Windows, etc.).

It should be noted that the UISRM addresses window based graphical user interfaces (GUI) and does not represent character mode or block mode types of user interface. It is however obvious that GUIs will be the prevailing types of user interfaces in the future.

Table 4.1 User Interface System Reference Model

	Layer Name	Purpose
6	Application	Provides the functions of an application (e.g. the content of the Application Layer).
5	Dialogue	Specifies the dialogue between the user and the application by associating user actions (e.g. mouse clicks, etc.) with application actions.
4	Presentation	Specifies the appearance of the UI, e.g. the "look" of the objects.
3	Toolkit	Specifies a set of UI objects (e.g. menus, push buttons, scroll bars, etc.) to be used in building an application's UI.
2	Subroutine Foundation	Specifies the primitive required to build UI objects (e.g. windows, buttons, e.g.) such as creation and destruction of objects.
1	Data Stream Interface	Specifies a call interface (function library) converting data from the application into the defined data stream encoding. This layer may also perform error handling and synchronization.
0	Data Stream Encoding	Specifies a network protocol defining the format for the data exchanged between clients and servers. The specification of the layer is independent of operating system, programming language and network.

4.3 REFERENCES

[4.1] Hoegberg, K. T., "Recent Experiences Related to the Implementation of an Open Systems Architecture", AFCEA Turkiye Seminar, Ankara, 1993.

CHAPTER 5

SYSTEM ARCHITECTURAL DESIGN

5.1. INTRODUCTION

Having defined the Reference Model (RM)* which, as stated previously, is the primary tool for expressing functional and cost related goals in technical terms we now come to the second step of the design methodology , which is the architectural design and this comprises the process of selecting available technologies for each of the functional elements in the RM ranging from specific products to international or de-facto standards.

For this to be done we must, first of all, define technically the system architecture in terms of its functional elements. From a system level point of view a C^3I system can be described as a distributed information system composed of node (associated with user locations) information systems (shortly node) dispersed throughout the organization, interconnected through Wide Area Networks and "coupled" by means of the following data exchange services (a part of Common Services in RM) [5.1]:

- Message Handling,
- File Transfer and Access and
- Remote Database Access and Replication.

The topological model of the system will be as shown in Figure 3.2. The technical description of the system level architecture which will be dealt with in this chapter will start with the "node level architecture" followed by first, the wide area network interconnecting the nodes and second, the "common services" which will be used for message and data exchange between the users. Finally we shall deal with the standards applicable to nodal and network elements including the criteria for the selection of these standards. The system architecture that emerges at the end of this design step will be transformed into actual hardware and software in the third design step which we call "System Configuration". This step will be dealt with in Chapter 6 which will cover, among other things, the selection of products in accordance with the architecture and development of the functional area dependent (system specific) applications required to realize the user needs.

---

* Definitions of the abbreviations used in this Chapter are found in Annex 5-B.

5.2. NODE LEVEL ARCHITECTURE

This section describes the C^3I system at the node level. The nodal architecture is first described logically, i.e. in terms of the organisational entities and the information flow among them. Then the node system is described in physical terms, addressing the system components, the data management and exchange and the applications.

5.2.1. Node System Logical Structure

The logical structure of the node system comprises the main organisational entities typical of a military or a corporate headquarters and the information flow among them. The model (depicted in Figure 5.1) distinguishes the following User Groups:

- Functional Area User Groups, e.g. Operations, Logistics, Intelligence etc.
- Command Group Users
- The Communications and Message Centre Group
- The System and Security Management Group

Functional Area Group Users are responsible for the parts of the command/corporate databases that are specific for the Functional Area. The responsibility includes control of access, update and, in general, maintenance. They may also require access to other Functional Areas databases. They need also exchange data with other Functional Areas and possibly with external systems through special interfaces. All users of Functional Area Groups require Common Application support (such as word processing or briefing support) as well as Special Applications (such as "logistics management, situation assessment") which provide functional area and task specific support.

Command Group Users require information from the Functional Area Groups. This information is typically tailored for the Command Group in the form of briefing material or other prepared forms. Tasks and decisions issued by the Command Group need to be submitted to the Functional area user groups for execution. Means for monitoring task execution are also required. Command Group users require Common Application as well as Special Application support (mainly decision support tools).

Communications and Message Centre staff are responsible for supporting the headquarters with various communication and information services such as:

- maintaining the connectivity to the outside
- managing the formal message (reporting) and informal message (e-mail) flow

- providing the central registry, distribution and directory services.

System and Security Management Group staff carry out all the tasks necessary to keep the node system operational including:

- system administration services
- operating services such as back-up/recovery, restart, error handling, system monitoring, resource administration
- help desk functions
- security administration services
- data and database administration
- WAN and LAN networking administration

Interfaces to External Systems can be common, i.e. relevant for several user groups, or they can be special, i.e. dedicated to one specific functional area user group.

Users operate from central, regional or local headquarters facilities depending on the situation and on the arrangements. These facilities can be located in different sites or co-located.

5.2.2. Node System Physical Structure

The C^3I node-system architecture follows the principles of distributed and open systems. It is based on the "client/server" model. These principles are commonly accepted in commercial information systems and in modern military C^3I system developments as explained in Chapter 1..

The Node system can be defined as a distributed information system composed of hardware and software components, located within a user (Headquarters) site, interconnected through one or more Local Area Networks, and "tightly coupled together".

This approach is intended to provide the flexibility needed to evolve with the evolution of technology, to allow growth or increased performance through the addition of resources, to improve reliability by adding redundant processing elements, and to minimise costs by allowing competition among a large number of hardware and software vendors.

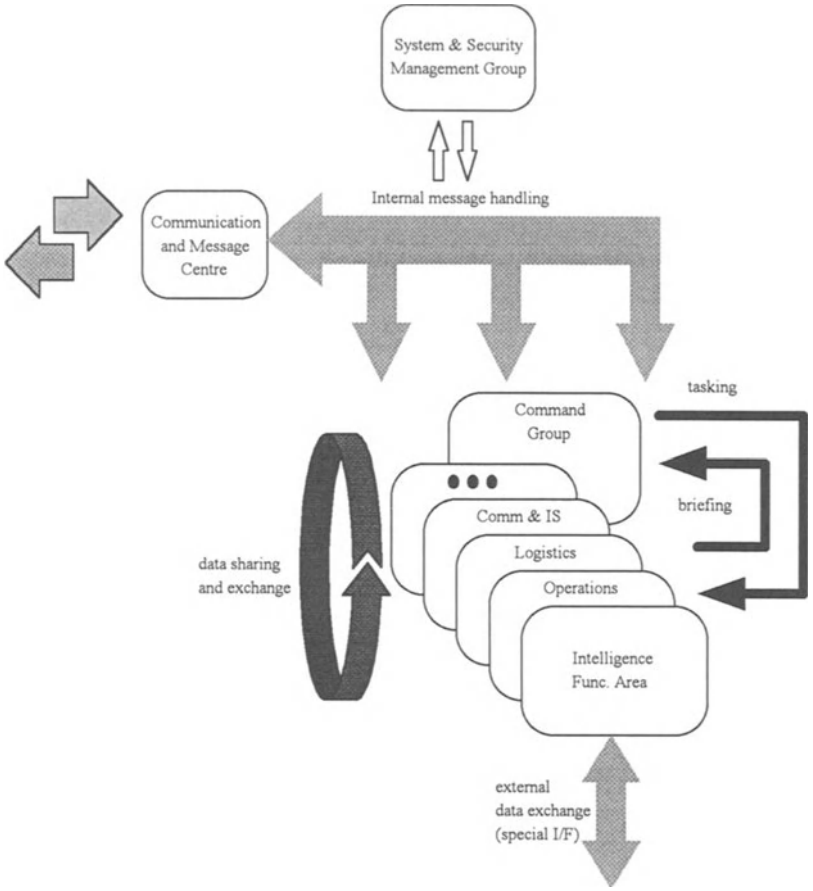


Figure 5.1 Logical Structure of a Node System

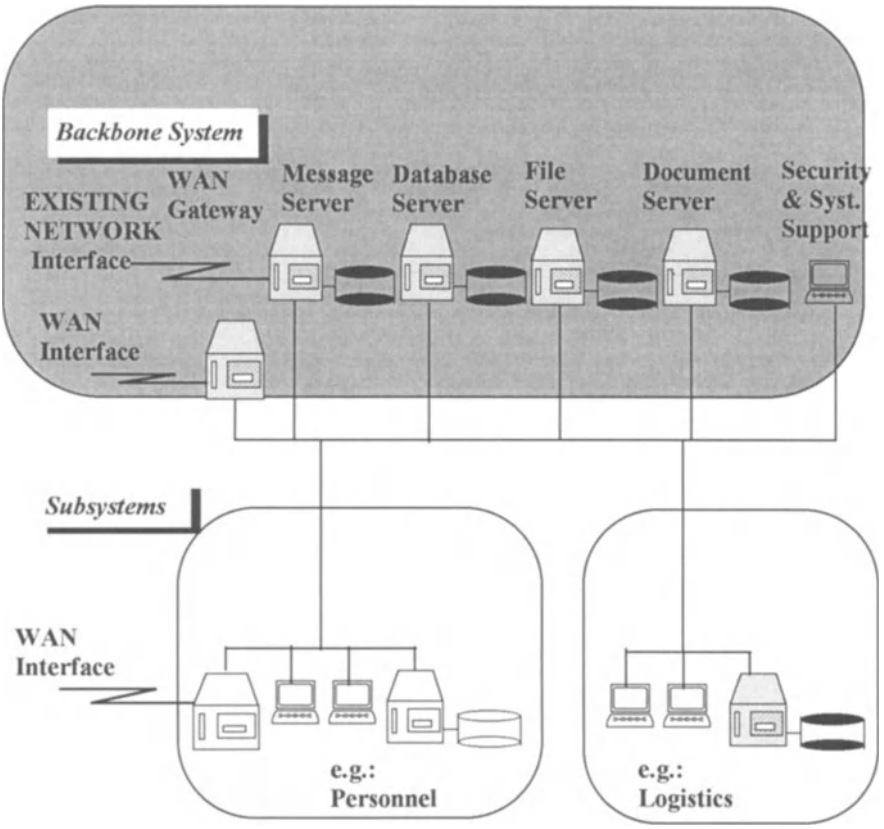


Figure 5.2 Physical Structure of a Node System

Figure 5.2 depicts the physical structure of the Node System. The "client/server" model (at this level of detail) comprises intelligent workstations (clients) and multi-user processors (servers) communicating over the Local Area Network (LAN) through open system protocols. This approach favours the transparent distribution of logical resources among various physical components thereby enabling a high degree of flexibility and expandability of the whole node system. It also permits the use of heterogeneous equipment.

In the node system architecture two levels are distinguished:

- the **Backbone System**, comprising the components that provide central node system (headquarters-wide) services,

- the **Sub-systems**, providing support for user-groups. A user group can be formed by the C^3I system users within a functional area, or within other organisational or physical elements of headquarters.

A subsystem is implemented around a sub-system LAN connected to the backbone LAN, the type of connection depending on the LAN technology and the degree of security control required to protect functional area subsystems.

Sub-system configurations can range from a number of workstations linked to the central services of an own client-server network consisting of workstations, servers, and possibly other special devices (e.g. a large screen display for command-briefings). Functional Area servers may comprise a Functional Area database server, file server or gateways to provide special interfaces to external systems. The exact configuration is based upon user requirements and finalised during the implementation process.

The concept of sub-systems provides the flexibility to develop or procure sub-systems for different functional areas or user groups independent from each other. Each subsystem can be supplied by a different vendor and may at some point in time be of a different technology generation. The overall system integration will be ensured by sharing the central services, and by conforming to the standards set by the central services.

The node level architecture specifies a range of components and services that can be tailored to the actual headquarters configuration depending on the local conditions and requirements. Configurations can range from systems for a large static headquarters complex to a small mobile C^3I system.

Security aspects are addressed in detail in Chapter 7.

Remote User Groups (i.e. users associated with a command or corporate headquarters but located remotely from the main node-system) can be supported with a simpler configuration by providing them with means to access remotely the main node backbone services. Figure 5.3 illustrates a typical configuration.

5.2.3 System Components

The components providing the Node System physical structure are summarized below.

5.2.3.1 Servers

The **Message Server** will be implemented using X.400 related standards (civil MHS, MMHS) The WAN connections will be transparent to the users. The main functions of the message server are central message handling functions such as message control, checking, registration, repository, distribution, directory. The message server provides standard interfaces (ISO and industry standards) to workstations and subsystems.

The **Database Server** is the component that supports the storage, control, distribution and allocation of structured data (text and numerical information). This definition corresponds to the functionalities supported by most commercial DBMS.

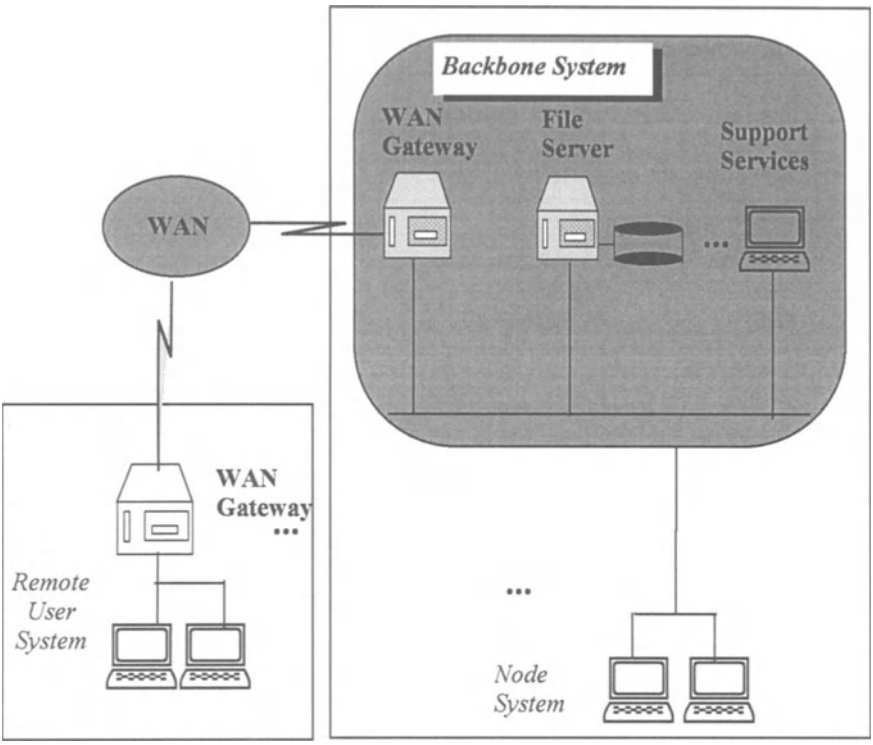


Figure 5.3 Remote User Group Configuration

The **Document Server** is the component providing support for the storage, control, distribution and information retrieval of complex and possibly large data objects (documents including text and graphics, voice, maps, charts and, in general, multimedia objects).

The **File Server** component supports the management of shared data outside the control of the database or document server. Access to the data on one or multiple file server will be transparent for users and applications within the node.

5.2.3.2 Workstations

The Workstations are the main resources to run applications (both Common and Special). The workstations execute most of the highly interactive services (most demanding in terms of response times) downloading some computational workload to the server (following the client/server model). Depending on the user needs, workstations of varying capacity will be used, ranging from high-end workstations with multi-media capabilities and large storage to low-end personal computers implementing only a sub-set of the complete functionality. Workstations software will include the User Interface and Graphics services, the interfaces to the server

and the LAN and the Common and Special applications that are needed at the specific work position.

5.2.3.3 Local Area Networks(LANs)

The Local Area Networks (LANs) connect all components and sub-systems in a node. A general networking structure of a node system comprises a backbone LAN which inter-connects the central components and the sub-systems, and several LANs dedicated to sub-systems. The sub-system LANs connect workstations and servers within the sub-system. In order to select a physical networking configuration, specific requirements (e.g. number of connected workstations, anticipated load, security) and expansion expectations are to be considered.

Just over fifteen years ago, Local Area Networks (LANs) were not considered a serious computing environment; mainframes and hierarchical Wide Area Networks (WANs) dominated the networking scheme. In the early developments, the main problem addressed by LANs was the connection of synchronous terminals to mainframe and minicomputer hosts. LANs represented an alternative cabling scheme. Both the impacts of LAN standards and PC's rapid migration into the workplace were coming into the scene. Later the IEEE 802 series LAN standards efforts became a family of standards and PC LANs, using PC as a file server became a common practice [5.2].

LANs have then emerged as essential links for internetworking computers. The emphasis has shifted to interoperability of multivendor systems encompassing multiple LANs and WANs. In comparison to a WAN, LAN spans a limited distance, often within a single building or group of buildings. The LAN may use ordinary twisted pair wire from each device that may then connect to a backbone coaxial cable between the floors, and possibly a fibre link between buildings to interconnect terminals, computers and peripheral devices. Users share access to common resources, information and peer-level communications. For wide area networking, connections among LANs may be established using interconnect devices such as bridges and routers in conjunction with leased lines or digital services.

Connections to the LAN are achieved by interfaces or adapters that plug into each device on the network, enabling them to communicate with other devices on the LAN. These interfaces may be pooled at a protocol converter (i.e. LAN server or gateway) to access other LANs or computing environments that use different protocols. Special-purpose LAN servers are available to facilitate resource sharing and protocol conversions between different computers and LANs.

Although the LAN concept has been well-defined since the mid 1970s, early developers had no universally accepted standards to use. Without such documentation, each vendor developed its LAN a little differently. Several approaches have been experienced on topology of the system, access methods used to transmit data on the network and transmission medium used for the LANs.

All these aspects of LAN are treated in Annex 5-A from which a LAN with appropriate characteristics may be selected to suit the particular needs of the user. There are several publications treating LAN's of different capacity with features such as topology, access methods and transmission medium [5.2, 5.3, 5.4, 5.5].

5.2.3.4 LAN Interworking Devices

Most business corporations use Local Area Networks (LANs) to support office functions and business-specific applications. As corporate structures slim down into flat, peer-to-peer relationships, sharing files and communicating all kinds of information across diverse networks becomes necessary. This need to connect LANs to other LANs or LANs to WANs may arise from normal business expansion or as the result of a corporate merger or acquisition. Whatever the justification is for linking dissimilar networks, it is becoming more popular and so is the devices such as Repeaters, Bridges, Routers and Gateways that are designed to do the job [5.2, 5.6, 5.8, 5.9, 5.10].

Interconnecting LANs often depends upon existing computer and communication networks that are often installed with little thought to this possibility. However, the best machine is often chosen to fit the application. The corporation computer resources grow without a single computer supplier providing any maintenance support despite the problem of incompatibility among different computer vendor's equipment. This situation has accelerated in recent years as the mainframe-dominated communications architectures of the 1960s and 1970s gave way to the LANs and WANs of the 1980s and 1990s. Moreover, the major computer vendors themselves each became identified with a particular LAN topology-DEC with ETHERNET, IBM with token ring, AT&T with StarLAN, among others.

Today's corporation comprises specialized work groups, and a variety of networks may be in place to meet their differing needs. The interoperability of these work groups often is limited by proprietary computer communication architectures that favor different types of LANs. The research and development division of a large company, for example, may have chosen an ETHERNET LAN to support its DEC net applications; the business group, IBM's token ring; and the technical documentation group, Apple Talk as the means for linking Macintosh microcomputers used for publishing applications.

Internetworking begins when the groups eliminate duplication of effort by drawing upon the resources of the others, improving efficiency and productivity. The connection of neighboring LANs spans to widely disperse work groups and, finally, the overall enterprise. The enterprise network may stretch around the globe, connecting a corporation's internal operations, customers, and suppliers. The devices that feed traffic on these networks fall in the categories of repeaters, bridges, routers, and gateways.

These devices all interconnect networks, and to various degrees their functions overlap. So the choice of interconnect device may not always be clear. Their proper use requires delving into the layer functions of the OSI reference model. In this communications model, every layer has its own set of protocols that provides a set

of services to the adjacent layers. The reference model consists of a suite of communications protocols at each layer that perform the myriad tasks required by the communications environment [5.18]. A layer on one device communicates with its peer on another device after the message has passed through the various layers.

Layer 1- The lowest layer is the physical layer, which is concerned with the transmission and reception of raw bits to and from the physical media used for communication.

Layer 2- The data link layer provides for the errorless transmission and reception of frames of information. This layer defines a fixed communications path across a network based upon device addresses.

Layer 3- The network layer or internetworking layer specifies the network topology based on user-defined network addresses.

Layer 4- The transport layer provides end-to-end error protection and, with the lower layers, forms a logically seamless data pipe for the upper layers.

Layer 5- The session layer establishes computer communications for a specific task; for example, mail, file transfer, and database query.

Layer 6- The presentation layer accommodates the way that data are represented; for example, ASCII or EBCDIC character sets.

Layer 7- The application layer completes the communications for the application as defined by its user.

a) Repeaters

A repeater represents the simplest type of hardware component in terms of design, operation, and functionality. This device operates at the physical layer of the ISO Open Systems Interconnection Reference Model (OSI/RM) regenerating signals received on one cable segment and then retransmitting them into another cable segment. Figure 5.4 illustrates the operation of a repeater with respect to the ISO OSI/RM [5.8]

There are two basic types of repeater. An electrical repeater that simply receives an electrical signal and then regenerates the signal. During the signal regeneration process a new signal is formed which matches the original characteristics of the received signal. This process is illustrated in the lower portion of Figure 5.4. By transmitting a new signal, the repeater removes any previous distortion and attenuation, enabling an extension in the permissible transmission distance.

The second type of repeater that is commonly used is an electrical optical device. This type of repeater converts an electrical signal into an optical signal for transmission and performs a reverse function when receiving a light signal. Similar to an electrical repeater, the electrical / optical repeater extends the distance that a signal can be carried on a local area network.

OSI Operation

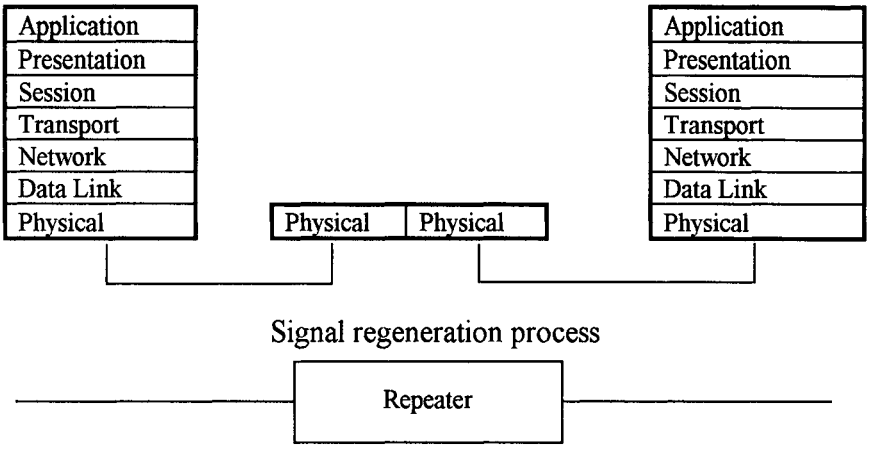


Figure 5.4 Repeater Operation

Repeaters provide entry points for the network manager. Repeaters are also taking on the added functions of linking different types of network media., fiber to coaxial cable, for example. Often LANs are interconnected in a campus environment by repeaters that divide the LANs into connected network segments. The segments may employ different transmission media; thick or thin coaxial cable, copper twisted-pair, or fibre.

As the traffic load on a LAN increases, an extended LAN may get overloaded under normal use. This occurs because a larger number of users must contend for the same amount of bandwidth. The slow response can be quite frustrating to the user who merely wants to send a message or print a document. When enough users are adversely affected by poor LAN performance, the situation can be devastating to the organization that relies on the productivity of its members for survival. A more intelligent device is required to extend the LAN while keeping the segments isolated. The device used for this purpose is a bridge.

b) Bridges

Bridges are OSI layer-2 components used to interconnect a sub-system LAN to the backbone LAN. A bridge allows isolation of the traffic within the sub-system LAN from the rest of the system.

The bridge provides network extension or interconnection for LANs, reading the individual LAN frames on one segment and only routing the frames between segments that are addressed to other segments. A bridge connects LANs at a relatively low level, the Media Access Control (MAC) sublayer of the data link layer. It routes by means of the Logical Link Control (LLC), the upper sublayer of the data link layer. Most often, it connects LANs of the same type, but some bridges are available to interconnect ETHERNET and token ring LANs. The distinctive

feature of a bridge over any other intelligent connection device is the connection speed, which could exceed 30,000 packets per second (pps).

Interconnected LANs look like a single LAN to attached devices. The bridge accomplishes this by performing three tasks:

- It keeps local traffic on the LAN, while allowing interLAN traffic to be routed between LANs. By forwarding only frames addressed to devices on other segments, bridges increase the throughput of the LAN. All electrically isolated segments connected by bridges form a single logical network. These networks may be very large, consisting of thousands of devices.
- It learns the device addresses on the LAN. Device MAC-layer addresses are assigned by the IEEE. Each manufacturer of LAN-attached devices must embed a unique address within the device. The MAC-layer addresses are permanently assigned to attached stations in a flat addressing scheme. The bridge-routing table stores the full 48 bit address of every station. When a learning bridge is attached to a LAN, it spends several seconds reading the frame addresses and storing them in the routing table. These addresses uniquely define each device attached to the LAN.
- It learns the routes between LANs. Bridges do more than link local LANs. Because bridges know local LAN device addresses, bridges can link LANs at remote locations using standard routing algorithms. Since bridges operate at the data link layer of the OSI model, they are transparent to the higher layer protocols. Thus, bridges can send traffic involving incompatible protocols across networks.

Some of this functionality goes beyond what was originally envisioned for the data link layer and, in fact, was developed by the IEEE independent of ISO standards. Eventually agreement was reached by the various standards groups, but not before the data link layer was split into two sublayers (Figure 5.5).

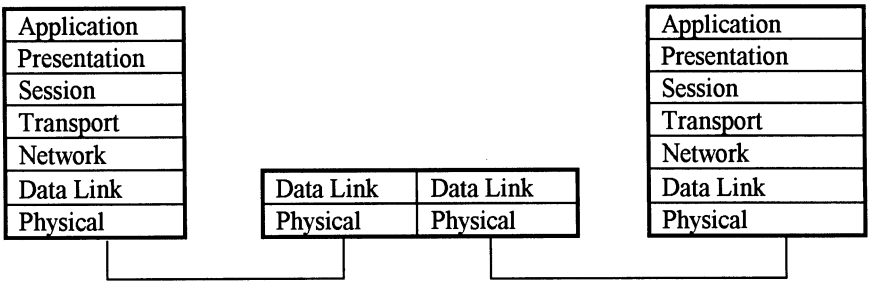


Figure 5.5 Bridge Operation

The MAC sublayer specifies how a device transmits and controls the signal over transmission media ranging from coaxial cable, twisted-pair wiring, fiber, and radio frequency. A series of medium-dependent access control methods is standardized by the IEEE, including CSMA/CD, token-ring, and FDDI. The upper

sublayer, the LLC, adds the routing capability to the data link layer. It establishes connections, transfers data, and terminates connections. Three types of flow and error control services are provided.

Type 1) **Unacknowledged connectionless service** is the service usually associated with intraLAN transport. There is no logical connection between the source and destination prior to data transmission. The frames are delivered on a best-effort basis, employing datagram service. So there is no guarantee of delivery, and it is the responsibility of the receiving device to request retransmission of lost frames.

Type 2) **Connection mode service** uses a logical connection between source and destination that is established prior to data transmission. This service is often used for interconnecting LANs because it relieves higher level protocols of connection management, while providing an efficient method for lengthy data exchanges.

Type 3) **Acknowledged connectionless service** further removes the burden of connection management by acknowledging frame receipt, a task usually associated with the transport layer of OSI. By providing this function at the data link layer, a large number of limited intelligence devices may communicate with a central processor. This type of service is useful in point-of-sale (POS) or factory assembly floor applications.

c) Routers

Routers are OSI layer-3 components to implement gateways between LANs or between LANs and WANs up to layer 3 [5.6]. The first case may happen to connect a LAN to an existing system. Routers can be used to implement the WAN Gateway component described below.
A router joins networks at the network layer (Fig. 5.6). It is a protocol dependent device that distinguishes among different communication protocols and applies the appropriate routing technique to each. With the requirement to interconnect a growing number of stations, servers, and hosts over multiple network, routers play an important part in holding these networks together, enabling them to operate and be managed as an enterprise-wide utility. Routers may be used to build huge, complex internetworks that rely on the network layer for efficient packet transmission. In fact, the router architecture is that of a packet switch, connecting multiple LANs and WANs. At the packet level, a router is readily compatible with WAN packet switching like X.25. In the process, routers offer the highest degree of redundancy and fault tolerance.

There is little physically to distinguish a bridge from a router. Routers and bridges are part hardware and part software. A bridge and router could be based on the same hardware platform, but routers join networks at the network layer, making them more processing intensive. For a router to perform the same function as a bridge at the same speed, innovative hardware designs with parallel processors or RISC architectures are required. This makes routers more complex and costly than bridges.

In return, routers have the ability to create many different subnetworks within an internetwork. These networks can be potentially independent administrative domains that provide a more manageable network. The router may keep a map of the entire network. It uses the map to examine the status of the different paths to the destination so it can determine how best to get the packet to the addressee.

The network layer protocol has its own source and destination information with which the router determines how to transmit packets to their destinations. The Internet has systems of networks and routers called autonomous systems (AS). Within an AS, the routers communicate routes that relate to network connectivity. Within a particular network, the routers keep track of host addressed, only routing on the host address if the source and destination are on the same network.

When a packet arrives at the router, it is held in queue until the router finishes handling the previous packet. Then the router scans for the destination address and looks it up in its routing table. The routing table lists the various nodes on the network as well as the paths between the nodes and their associated cost. If there is more than one path to a particular node, the router will select the most economical path. If the packet is too large for the destination network to accept, the router segments it into several smaller packets, a process referred to as fragmentation in TCP/IP terminology but also known as translation. The capability is especially important in adjusting to WANs. With smaller packets, there is less chance that noise or other line impairments will corrupt the data. Even if that occurs, the error can be detected and a retransmission requested. For store-and-forward type packet networks, smaller packets actually results in higher throughput when frequent retransmissions occur. Consequently, public packet-switching networks have standardized on 128 byte packets rather than on 1,500 byte frames used on ETHERNET LANs or the 576 byte packet standard used by internet.

A major difference between bridges and routers is the way frames are handled. A bridge forwards all MAC-layer frames. Routers, in contrast must be programmed with software that is specific to each network protocol to be forwarded. Also routers can use multiple paths between any two points, but bridges can support only one logical route between any two points.

Routers are very good at bypassing link failures and congested nodes, which is critical for applications that can not tolerate unnecessary delays, prolonged outages. Bypass is facilitated by the ability of routers to share information with one another. Bridges can not do this, because they do not have access to the OSI network layer protocol. Thus, when one bridge gets overloaded, the others will never know about it. Packets may simply be lost unless the end devices have the intelligence to request the retransmission of missing packets.

Despite the differences, the task of the router is similar to that of the bridge; to identify the devices in the internetwork, set up the paths among LANs, and determine the criteria for data transport.

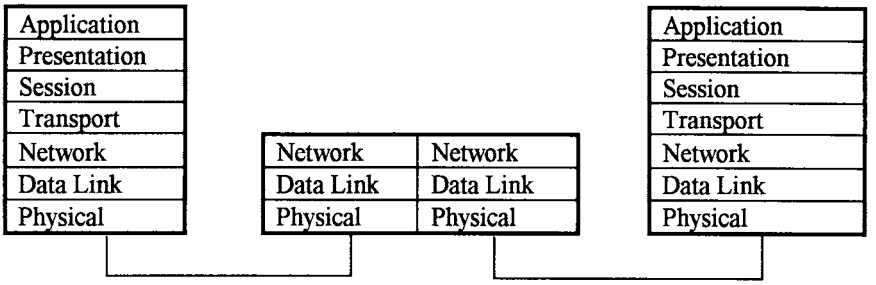


Fig. 5.6 Router Operation

d) Gateways

The WAN Gateway component provides the interfaces to the Transport Area network, including existing X.25 networks, and to standard PTT communications [5.6]. The interface to the existing network is provided by the Message Server. Normally the WAN Gateway provides an OSI layer 3 networking service. Although not part of the gateway, the crypto equipment is required for operation of a system high classified network and is included with the gateway.

Because organizations are comprised of specialized work groups, a variety of networks may be in place to meet the different requirements of users. For example, the research and development division of a large company may have chosen an ETHERNET LAN to support its DEC net applications, while the technical documentation group may have chosen Apple Talk as the means of linking Macintosh microcomputers to support its technical publishing operation. The two groups discover that they can eliminate duplication of effort by drawing on the resources of the other. A server equipped with both ETHERNET and Apple Talk circuit boards perform the necessary protocol conversions that allow users on both networks perform the necessary protocol conversions that allow users on both networks to exchange files. A device that performs protocol conversions that allow information to be exchanged between two different types of networks is called a gateway.

A gateway encompasses the functionality associated with all seven levels of the OSI Reference model (Figure 5.7). Gateways interconnect networks or media of different architectures by processing protocols to enable a device on one type of LAN to communicate with a device on another type of LAN. The gateway acts as both a conduit over which computers "speak" and as a translator between the various protocol layers.

As corporate divisions become more interrelated, the need to share files and communicate all types of information across diverse networks becomes necessary to improve efficiency and productivity. The need to connect dissimilar networks may also come about as the result of corporate merger or acquisition activities. Whatever

the justification for linking dissimilar networks, gateways are designed to do the job.

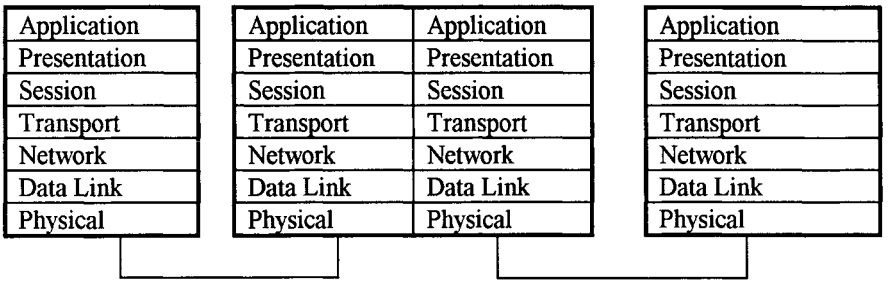


Figure 5.7 Gateway Operation

Gateways are more capable than bridges or routers because they not only connect disparate networks, but they also ensure that the transported data from one network is compatible with that of the other. Not only does this higher level of functionality translate into a higher price for equipment, but the translation function imposes a substantial overhead burden on the gateway, which results in a relatively slow throughput rate (hundreds of packets per second versus tens of thousands of packets/s for intraLAN or remote bridges). Consequently, the gateway may constitute a potential bottleneck when utilized frequently, unless the network is optimized to mitigate that possibility.

Access to the gateway may be controlled by assigning specific ports to certain microcomputers. When a microcomputer requests access to the gateway, it is given the port reserved for it. Because no other microcomputer can access to particular privileges. One port may provide access to all mainframe applications, for example, while another port may be limited to only one application. The problem with dedicated access is that idle ports can not be used by anyone else, which means that efficiency is sacrificed for the sake of security.

When security is not an issue, gateway access may be provided on a contention basis. This arrangement provides more opportunities for users to create links with the mainframe or other network resources because port contention does not limit users to specific gateway ports. Some gateways permit both shared and dedicated access, allowing some ports to be reserved for specific microcomputers and the rest pooled for general use.

When a separate server is used as a gateway, saving result in cabling costs and installation time, and moves and changes are easier to accommodate. In fact, users can change the physical location of their equipment, but retain their logical address on the network. With communication functions off-loaded from the host in this way, valuable processing resources are freed for more important tasks.

Another advantage of the gateway is simplified network management. Instead of having to monitor the traffic from 100 microcomputers on the network, for example, only the traffic from a single gateway would have to be monitored, which

appears to the host as a single peripheral device. In this case, a separate cluster controller is unnecessary because the gateway replaces it.

Gateways can extract detailed information about the data traffic passing though it, as well as the status of the data links with which it interfaces. The gateway can ensure that the links are handling data reliably, without exceeding user-defined error-rate thresholds. The gateway can also check on the various protocols being used, making sure enough protocol conversion processing power is available for any given application. The gateway's management system can generate a variety of reports, which can be extracted automatically by time of day, or as demand warrants via keyboard command. Network statistics may be archived for trend analysis, which can assist in long-range planning.

In WAN environments, the gateway balances load levels, bypasses failed links, and finds the most economical route. With some gateways, all of these functions are performed automatically as the result of a single connect request from a user, regardless of equipment location or protocols involved. In this environment, the ability to detect, isolate, and diagnose problems becomes very important. The network management tools that are available with today's sophisticated gateways allow the remote configuration of channels, links, and other network interconnection devices such as bridges and routers. Through the network management system, gateway ports may be brought on or off line as required.

Because gateways perform protocol conversion, performance bottlenecks may become a problem. Every new connection, hop and protocol that is added to the network not only intensifies the problem, but invites new problem, such as higher system costs, limited growth and expansion, and non-transparent connections - all of which complicate network management. With so much networking overhead devoted to protocol translation, some gateways have become dedicated to specific applications such as E-mail and batch file transfer.

Some vendors are developing so-called "intelligent gateways". These devices are called gateways but operate more like routers. They talk to each other about the best way to route information, taking into consideration such things as congestion, priority, performance (throughput, delay, error rate), security, and even cost. Building such capabilities into intelligent gateways relieves users of having to make these decisions.

An inherent weakness of such schemes is congestion. Congestion may affect the performance of the entire network, or only one gateway of the network. Congestion may be caused by an inefficient routing scheme, causing traffic to stay on the primary data links longer than necessary and thus slowing down the entire network. Alternatively, congestion may actually be in the gateway, a situation that could occur when a gateway is presented with too many packets to filter. To minimize the chance for such bottlenecks, the gateway protocols must be able to perform flow control and respond to congestion indicators. End-to-end protocols such as TCP can cope with congestion. Finding out the cause of the congestion can make a difference in determining whether trying to reroute through other gateways attached to the network is worthwhile.

When congestion is detected, the intelligent gateway can prioritize the information that is to be routed; for example, the gateways can determine whether local or internetwork traffic should be given preferential treatment. Prioritizing information is also important in the management of the network. Intelligent gateways will have the ability to allow diagnostic information to pass through or around congested areas, providing real-time status reports on each link.

If the entire network is congested, then all of the alternative gateways located on the other site of the network can be bypassed entirely in favor of a hop through an entirely different network. In hopping through to another network to avoid congestion, security becomes a concern. This concern is addressed by the ability of some intelligent gateways to distinguish between routine and sensitive information during the routing decision.

5.2.3.5 Security Devices

Security Devices include encryption, access control, and data integrity equipment as well as associated management devices. Security will be addressed in Chapter 7.

5.2.4. Data Management and Data Exchange

This section describes the concepts for data management and data exchange used within a node system.

5.2.4.1 Database Distribution Concept

The term "database" is used as a general term to refer to a collection of related data, regardless of whether it is kept in files, relational tables or in other ways.

Data management also provides facilities to support the distinction among the following partitions of the operational database:

- live data: data used in support of live operations only.
- exercise data: data used in support of exercises only.
- training data: data used in support of individual or group training activities only.

Regarding the data distribution in a node system, the recommendation is for a compromise between a central database (with advantages on the side of integrity and consistency and administration) and a fully distributed database (with advantages on the side of availability, performance and security). Three levels of database (with structured and unstructured data) are distinguished within a node system:

- Command databases
- Functional Area databases
- User databases

The first type corresponds to the headquarters-wide support concept ("backbone system"), the second to the User Group support concept ("sub-system") and the third to the individual user support ("workstation").

- An integrated **Command Database** contains data of interest to the whole headquarters or to two or more User Groups (common data). It can be physically allocated across multiple backbone servers (database, file or document) for performance or availability reasons but it is managed as a (logically) centralised one. Data integrity, consistency and security control are enforced. This database should contain at any time the most current common data in a consistent state.
- Within each subsystem, **Functional Area Databases** may exist. They contain data specific to the Functional Area, i.e. for which the Functional Area User Group is responsible for the update. It is the headquarters interested in or to which it requires frequent access.

- On each workstation, **User Databases** might exist. Their management is completely under the control of the user or of the specific applications (both common and special) that support its tasks. User database management is not further considered here.

Groups of data (also called "partitions") in the Functional Area databases may correspond, totally or in part, to the data partitions in the Command Database. The distribution of the data at the sub-system level improves the performance, security, reliability and availability for the User Group but may cause some problem. The main problem is the possible consistency among the various copies of the data in the node system (in the Command Database or other subsystem databases). In order to reduce this problem, and improve the consistency of the data, the following design concepts should be implemented:

- Clear **ownership** of a User Group for each database or partition of it (e.g. file or table). Only the "owner" User Group is authorized to change this data and to establish the access right to it. A data partition can also be owned by an external entity., which means that no user group in the headquarters is authorized to update it.

- If a data partition is present in a Functional Area (FA) database for which the FA has no ownership, such a partition is a **replica** of the corresponding partition in the Command Database. A "refresh" mechanism is implemented to update the sub-system data with the most current node-wide version available in the Common Database. Various mechanisms can be chosen from an "immediate refresh" (i.e. done any time the data is changed), to a "refresh on demand" (i.e. done when it is read) or a "periodic refresh". Different mechanism can be chosen for different Functional Areas and database partitions. The choice of which is the most appropriate is left to the node system design considering the specific requirement.

- If the data partition can be updated within the sub-system (i.e. the Functional Area is owner), it is responsibility of the FA to update the corresponding replica in the Command Database. Similar mechanism to those previously described could in principle be implemented. However, it is only the Functional Area responsible staff who know when the partition is in a consistent state and can be shared at the node-level. An explicit "push" of the data (when believed correct and consistent) to the Command Database seems the most appropriate mechanism.

- The principles specified above can be implemented rather easily for structured data stored in relational DBMS utilizing features available in commercial off- the shelf DBMSs. The implementation for file-based data or document data is more complex and procedural, user-driven measures will be necessary.

- Various technical COTS solutions are available to improve the availability of databases and data such as: disk, file, database or server "mirroring", recovery logs, on-line back-ups etc.

5.2.4.2 Data Administration

The Data Administration function aims at maintaining and enforcing:

- common and consistent definition of all the data that is subject to exchange

- compatibility and consistency with definitions of databases in node systems.

- standard method for data(base) design and maintenance.

In order to fulfill these requirements, the concept of **Information Resource Dictionary System (IRDS)** is introduced:

- a standard method for structured data definition and database design. The definition language should be semantically rich. The Entity Relationship notation is recommended.

- a global (system-wide) logical and integrated definition for the common data (Conceptual Schema) following the currently applicable data definition standards.

- a Data Dictionary representing this definition, supported by an automated graphical tool to facilitate the definition, maintenance and validation tasks.

- an C^3I System Data Administration (DA) function to manage it.

Figure 5.8 illustrates the concept of an IRDS.

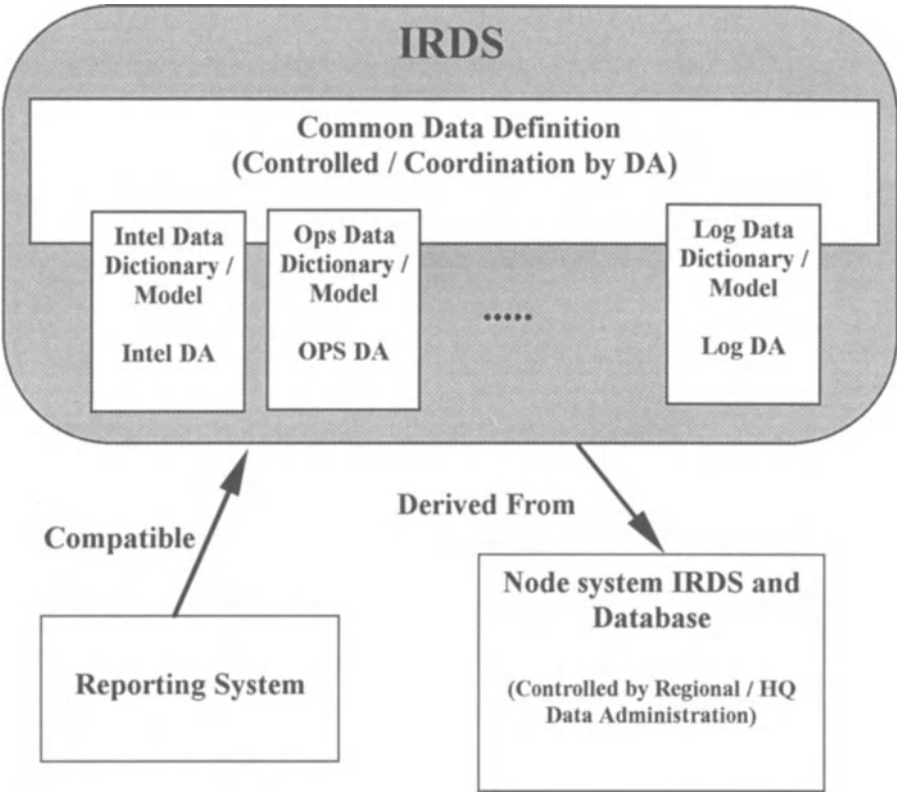


Figure 5.8 IRDS Concept

The IRDS is split into two parts :

- the **common part**, including all data definitions that are applicable across functional areas. This part can initially be populated with the existing (and still applicable) definitions.

- the **functional area specific part**, comprising data specific to one each functional area. Each functional area should define its specific part of the IRDS (guaranteeing the interoperability and consistency within the area as well as apply central co-ordination to ensure the minimum commonality needed for interoperability across functional area boundaries.

To implement the concept effectively, the following principles should be followed:

- The definition and maintenance of the IRDS will be a dynamic process as application prototyping and development efforts will lead to inputs and revisions of the IRDS. To manage this process, Data Administrator posts

and capabilities have to be installed both on the system level and on the Regional/Headquarters levels.

- The IRDS will serve system and application software implementors as standard reference for the definition and implementation of operational databases in each C^3I system node. Local (i.e. command specific) data definitions should be derived from and in consistence with the IRDS to the extent possible. The local database scheme should distinguish the command unique subset (with no interoperability requirements) from the standard definitions compliant with the IRDS.

- The IRDS definitions should be utilized by all C^3I system nodes when transferring structured data (in any form). The IRDS should be the basic tool for the definition and revision of any data interchange procedure, i.e. new messages as well as other types of structured data exchange such as files, database transaction formats.

5.2.4.3 Handling of Message Data

The OSE recommended standard for exchange of messages between node-systems is CCITT X.400 (ISO MOTIS). For node internal message handling the choice is between an implementation based on X.400 compliant products or an implementation based on industry standard E-mail solutions. Although X.400 based products are available for implementation of node-internal message handling, they can currently not compete in functionality, user friendliness, and integration with office automation environments, with commercial E-mail products, even though this situation is changing fast.

For the near term the following is therefore recommended:

- Use of industry standard COTS E-mail solutions for sub-systems.

- Use of X.400 as the common denominator at the backbone level in case sub-systems implement differing E-mail solutions or

- Use of a node-wide E-mail industry standard product with X.400 gateway

- Select a message handling/E-mail product which has integrated directory services based on X.500 or which at least provides an X.500 interface.

The implementation of message handling services within a node is based on one or several message servers and end user message handling applications in client workstations. The message server(s) provides (together with the WAN gateway) the gateway services for message exchange with the outside as well as all services for internal message exchange, handling and storage. End-user applications for message handling interact with the server providing end-users access to the central services. The backbone message server can directly serve workstations or it can serve sub-systems implementing their own subsystem message server. Figure 5.9.

illustrates a node system with X.400 based message server on the backbone serving two sub-systems which implement industry standard E-mail systems and users of the third sub-system directly.

Although a heterogeneous solution for message handling combining X.400 and industry standard E-mail is feasible, the effort for maintaining and managing a heterogeneous message handling system within a node is significantly higher than a homogeneous solution.

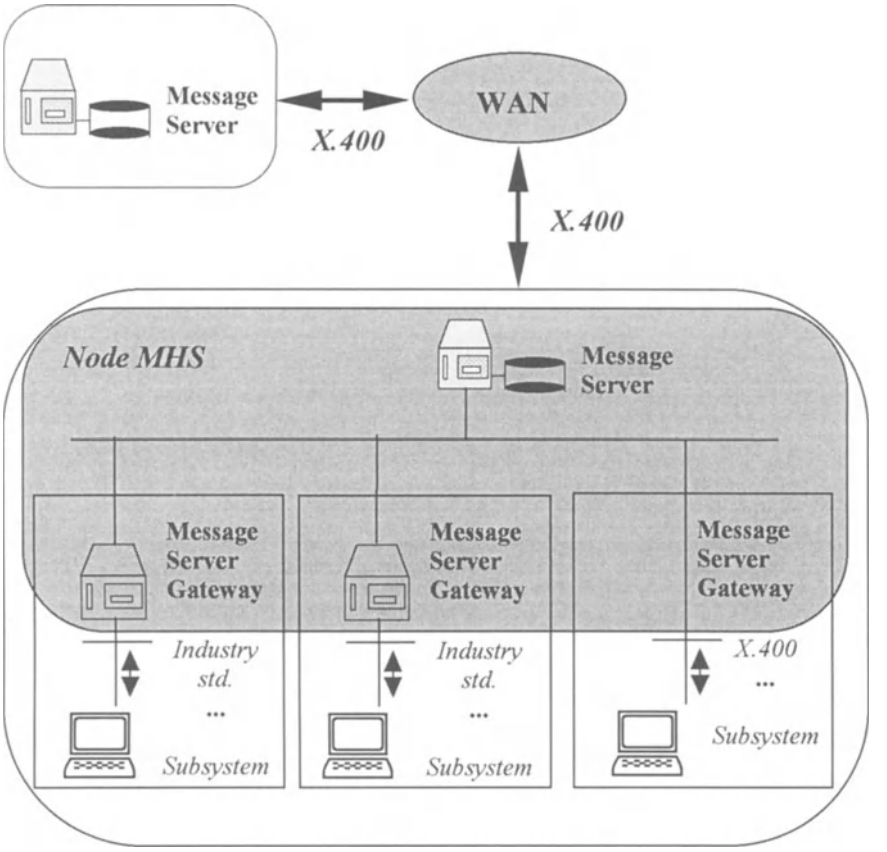


Figure 5.9 Node Message Handling

5.2.4.4 Handling of Document Data

Documents are considered to be objects such as memoranda, letters, forms and reports which may include text as well as graphics. Common document services include:

- import of paper and electronic documents either off-line or on-line from news-wire services or submitted from other organisations
- registration of documents
- distribution and retrieval of documents in accordance with user defined criteria like themes or topics.

Document services will be implemented by the document server on the backbone. In addition user-group specific services can be implemented by document servers within sub-systems. E-mail can be used to maintain the flow of documents from the central document server to possible sub-system servers, and finally to the end-users workstations. For on-line document retrieval functions special applications implemented on end-user workstations interact with the document servers via product specific protocols.

All such functions and capabilities can be implemented with COTS products. The main issue for implementing a headquarters-wide document service is the decision on standard document formats and interchange formats.

The relevant standard is ISO 8613. It is a multi-part standard including ODA, ODIF and ODL. Office Document Architecture (ODA) is an architecture that enables users to interchange the logical structure, content, presentation style and layout structure of documents from one application to another, or from an application to various output devices. Office Document Interchange Format (ODIF) is an encoding standard for document interchange. Office Document Language (ODL) is a Standard Generalised Mark-up Language (SGML) encoding for ODA documents to enter a SGML database or publishing environment. ODA document represented by SGML are interchanged using the SGML Document Interchange Format (SDIF, ISO 9069, 1988).

For C^3I system, ODA and the related standards should be seen as the mid and long term targets. As ODA conformant products are not widely available, near-term implementations will have to be based on proprietary products.

5.2.4.5 Handling of Files

The recommendation for FTAM for file transfer and access among nodes could be inefficient within a node system, where a more reliable and fast data transmission support is normally available. Also, FTAM has not yet been widely accepted and implemented in commercial products. The effort to integrate FTAM with COTS applications is considerable. For near-term implementations it is recommended to:

- implement the exchange of data files on the basis of E-mail services,

- use the distributed file handling capabilities either integrated in or supported by operating system products.

Combinations with industry standards for file management are feasible. Figure 5.10 illustrates a node system with two file management domains and the capability to share files between them.

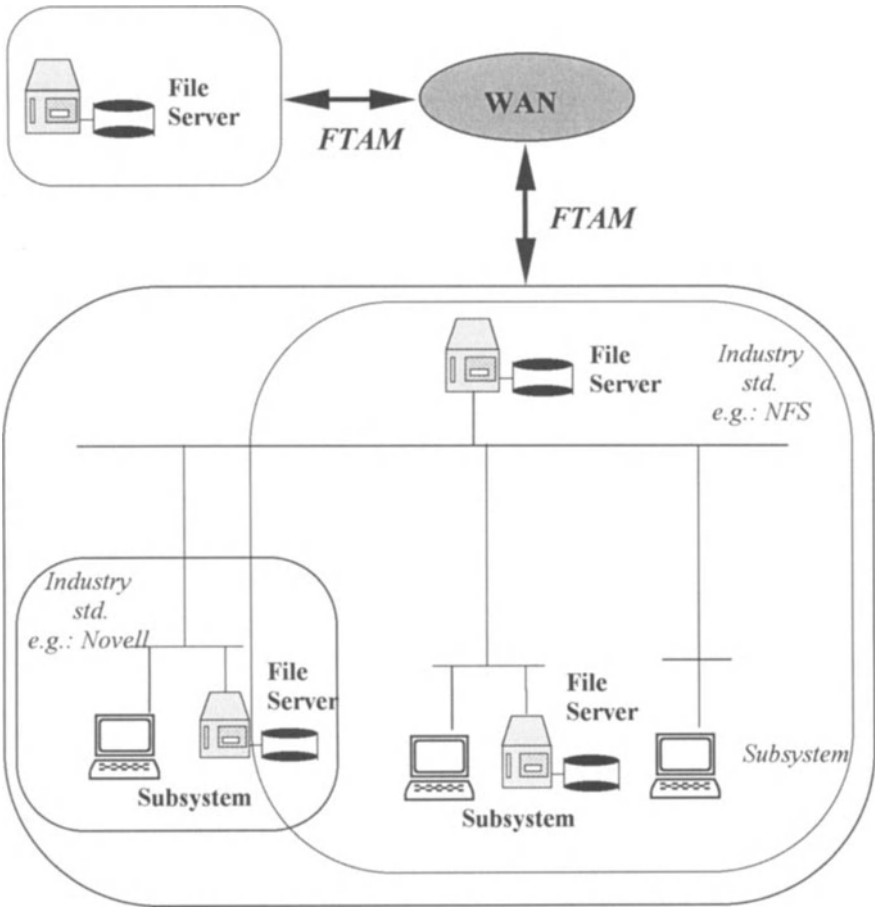


Figure 5.10 Node File Handling

5.2.4.6 Handling of Structured Data

The recommendation is for SQL and Remote Data Access (RDA, ISO DIS 9579-1, DIS 9579-2 (RDA-SQL specialisation)).

For near-term implementations RDA compliant products are not yet available. However, products offering RDA like capabilities do exist. An industry standard based on RDA is emerging (SGL-Access) and endorsed by X/Open. First SQL Access implementations are available and wide support by COTS products is expected.

Distributed database management in terms of application accessing multiple databases, and data replication among distributed database servers and data bases, is supported by some DBMS products. Implementation of such functions in a heterogeneous DBMS environment is not recommended for the near-term. It should also be considered that the effort of maintaining and managing a heterogeneous distributed DBMS within a node is significantly higher than for a homogeneous solution.

As illustrated in Figure 5.11 the use of different DBMSs at the backbone and sub-system levels implies the development and use of specific gateways if interaction between the databases is required.

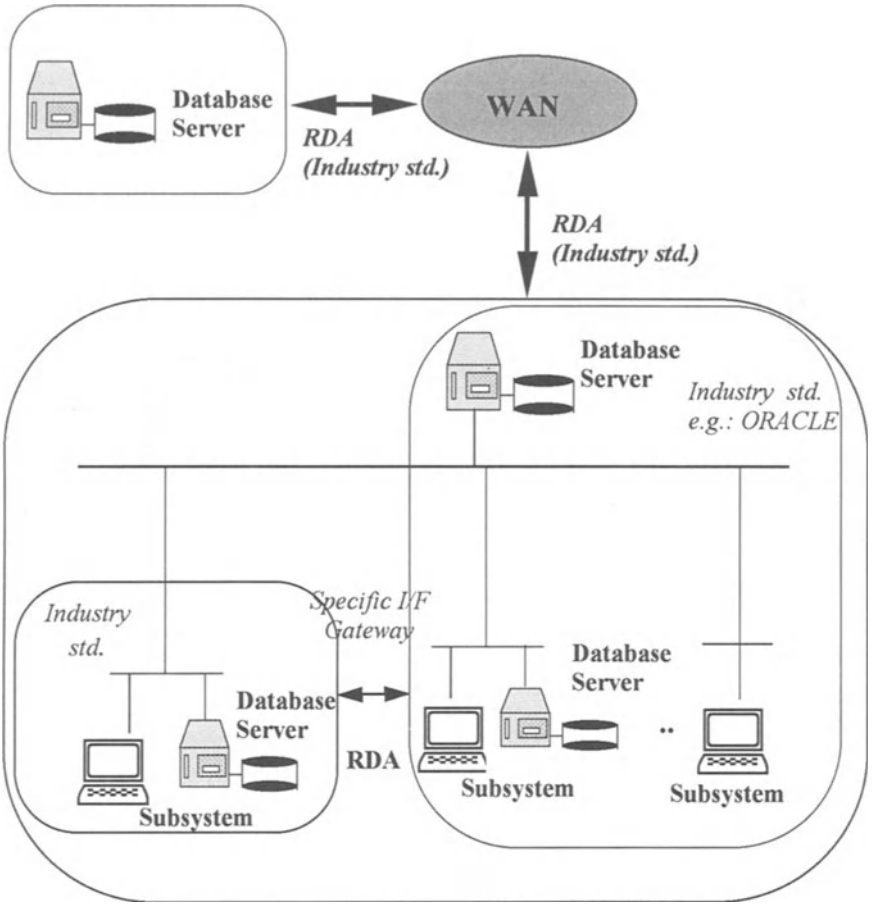


Figure 5.11 Node Database Management

5.2.5. Applications

The applications are parts of the C^3I system that are most visible to the end user. From their perspective (i.e. the "operational requirements") the function of "all the rest" of the C^3I system (more precisely called the "Application Platform" in the OSE Reference Model) is only to enable applications to be executed.

Two classes of applications are distinguished in the OSE Reference Model: Common and Special Applications. The next two sections deal with each of them.

5.2.5.1 Common Applications

The required Common Applications are described below:

a) Electronic Mail

Electronic Mail should be provided through COTS products supporting the exchange of informal messages within a node. A COTS package compatible with X.400 is preferred to facilitate E-mail and Message Handling integration. Integration between E-mail, word processing and business graphics packages will also be required.

b) Message Handling

An automatic desk-to-desk message handling system will be required, which includes the preparation and distribution of outgoing messages as well as routing and logging of incoming messages. Distribution of incoming messages should be based on user profiles that can be dynamically modified by users. Generation, co-ordination, revision, and controlled release of outgoing messages will be required.

Message Handling is described in Section 5.4 (See also [5.3]) with appropriate standards recommended. Message Handling can be provided through off-the-shelf packages running on UNIX platforms which are POSIX compliant. However, at the present time, such packages may include commercial X.400 implementations rather than STANAG 4406 which is expected to become commercially available later. It will be necessary to integrate message handling software with other packages so that documents or diagrams can be directly imported into message preparation. Message handling requires a significant effort to set up and administer with the necessary addressing and routing information, and packages to support directory services may be considered a good investment as the X.500 series of standards matures and COTS products become available.

c) Message Processing

There will be a requirement for message processing. The message processing will allow the automatic processing of incoming structured messages into database updates, and the profiling of outgoing messages and message preparation from the structured database.

Message Processing can also be supported through UNIX based COTS products, in some cases already integrated with commercial X.400 message handling capabilities, and using SQL for database definition and access. Glue code may be necessary for integration with other COTS products and a good deal of initial set up effort is necessary to define the messages from which database update will be done. At the system level, the functionality provided by message handling and message processing will be combined with services provided by the Man Machine Interface (MMI), Data Management, and Communications Modules to support both user-to-user and database-to-database interoperability. Figure 5.12 presents an overview of

the Message Handling / Processing with the required sub functions for database-level information exchange.

The automatic message handling/processing requires the messages to be formatted. For military C^3I systems, NATO AdatP3 Message Catalogue derived from NATO Message Text Formatting System (FORMETS) should be used for the formatting of messages.

As far as unformatted messages are concerned, these will be handled manually till the appropriate technologies such as automatic speech and text recognition, artificial intelligence become available. In the mean time, some semi-automatic tools may be used to speed up the manual operations.

d) Briefing Support

The generation of slides for briefing purposes can be supported by several COTS products including word processors, spread sheets, geographic and drawing packages. The organisation of a series of slides into a cohesive briefing with computer driven presentation through projection equipment or large screen displays can also be supported through COTS products designed specifically for this purpose. Integration of the packages which generate the slides with the package which organizes and presents the briefing will, of course, be necessary. Importation of pictorial information and production of hard copies can be supported through available hardware devices and associated software drivers. However, extensive use of optical scanners and digitizing equipment will impact on performance and demand more computing power in LANs, servers and workstations.

e) Significant Event Handling and Tasker Handling

Significant event handling can be supported through use of commercially available scheduling packages and E-mail/X.400 message packages.

f) VDU Conference

There are some COTS products which could support a VDU conference, such as standalone (ISDN) packages as well as desk-top integrated packages like Communique, Simplicity and Pro-share.

g) Document Handling

Document handling can be supported by word processing packages. Content related searches through large volumes of documents can also be supported through off the shelf Text Retrieval software.

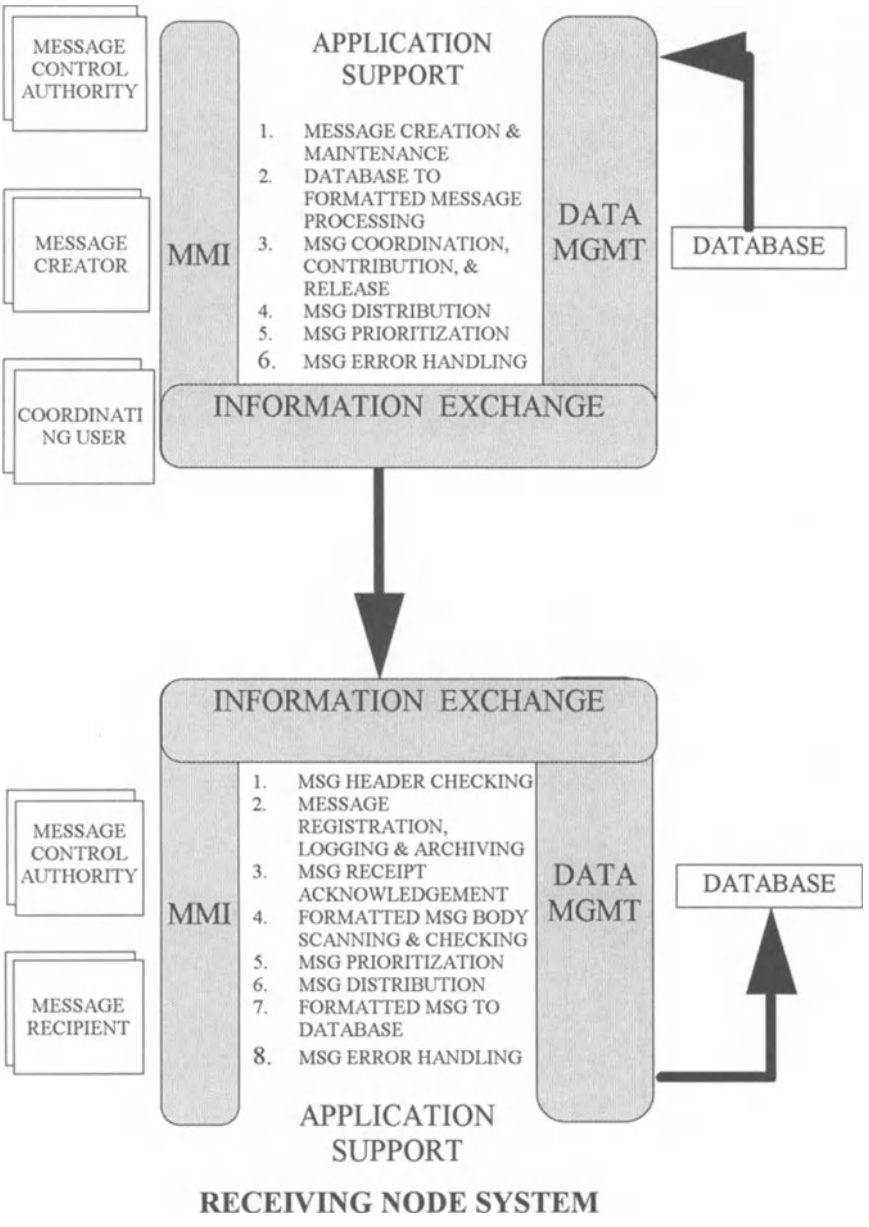


Figure 5.12 Message Handling and Message Processing

h) Data Views

Database queries, updates, reports and associated screen definitions are all supported by the 4GL facilities provided in the DBMS. In the case of this Architecture that would mean through use of SQL code. The combination of data retrieval and graphic or geographic display will require the integration of DBMS with other COTS products through the use of glue code.

i) Map Graphics

The capability to analyze standard symbols overlaid on maps or marine charts is also available off-the-shelf through the use of geographic packages in conjunction with off the shelf routines which can be purchased to calculate distances between co-ordinates etc. The use of output from the geographic packages in the briefing support software will depend on the import capability of the latter or the use of glue code.

j) Time Management

Time management can be supported through use of a scheduling package.

k) Office Automation

Office automation is, of course, available off-the-shelf in an integrated form. Use of database capabilities in other than an informal manner should be carefully controlled to ensure integrity of data.

l) Desktop Functions

Desktop functions are supported by the GUI package.

m) File Transfer

Database replication, high bandwidth data exchanges, including graphic and/or mixed data, order of battle updates and certain lengthy assessments will necessitate file transfer for which standards must be adopted for interoperability.

The file transfer functionality together with a database dump/restore functionality provided by data management, will provide a mechanism for bulk data transfer in support of the database replication.

The file transfer can be used in a heterogeneous environment; however to avoid a separate format-specific file translation application it would be necessary for the DBMSs at Node System to be compatible.

The OSI FTAM will be required for the File Transfer in a C^3I system, since it is more functional than TCP/IP FTP. In addition to providing a virtual storage capability, the FTAM user can transfer part of a file instead of the entire file, as is the case when using the FTP.

n) Remote Data Access (RDA)

C^3I database systems will be networked, implementing Client/Server Computing model. The Functional Area Subsystem (FASs) that are proposed to meet the computing needs of a functional area or a user group, will have its own specific database server. The FASs, needless to say, will require to exchange data with each other. Therefore, remote data base access (RDA) service will be required to establish a remote connection between a database client, acting on behalf of an application and a database server, interfacing to a process that controls data transfers to and from a database.

C^3I system RDA service will be provided in conformity with ISO 9579-1,2 standard. This is a generic RDA standard which provides a RDA service interface and an RDA communication element that exist at both the client and server sites. The generic service does not specify the syntax or semantic or database operations sent from client to server. C^3I database management system should be capable of supporting the proposed ISO based generic RDA.

The security functions provided within the RDA are similar to the security functions provided by a common SQL based relational DBMS. These are:

- Identification and authentication
- Verification of rights
- Auditing
- Object reuse
- Error control and correction
- Integrity

Unfortunately, the ISO RDA is not fully standardised and commercially widely available.

Most of the Common Applications can be implemented entirely or partially by COTS software packages.

5.2.5.2 Special Applications

In addition to the Common Applications, Functional Area specific applications ("special applications") and databases are required. It is expected that most of the Special Applications will be dependent only on the Functional Area rather than where in the organization FA is. These should therefore be applicable (and possibly portable) in different node-systems. Examples of Special Applications are:

- Order of Battle Management (Intelligence FA)
- Forces Deployment and Movement support (Logistics FA)
- Mission Planning support (Operations FA)
- Manpower Management (Personnel FA)

All the operating modes (e.g. peace-time, crisis, war and exercise) should be supported.

A more detailed definition of Special Applications is dealt with in Chapter 6 where analysis of existing applications and databases, testbedding and prototyping of new ones is recommended as the way ahead.

5.3.5.3 Application Software Development

Common Applications have a wide applicability by definition in all commercial and military systems. A large selection of COTS products is therefore available and no need is foreseen for developing Common Applications for C^3I systems. The availability of software for Special Applications as COTS products that suit the requirements of Functional Area users exactly is however more limited. When re-use of existing software is not feasible, there is a need to develop new ones. From the development point of view, three main types of applications are foreseen for the C^3I system:

- **database-oriented** applications, mainly dealing with the retrieval, presentation and update of data extracted from databases

- **graphics-oriented** application, mainly dealing with the presentation of graphical data and the management of the human-computer dialogue

- **computation-oriented** applications, including various types of applications among which the most relevant for C^3I system are decision support tools (e.g. simulation or analytical models).

Software complexity varies but, generally speaking, the first type is the simplest. From the analysis of existing C^3I systems database-oriented applications seem to support most of the requirements in a C^3I system.

The so-called "fourth generation languages" (4GL) are the most productive way to develop database-oriented applications software once the database structure has been designed usually using CASE tools. Since there are no standards for 4GL an appropriate selection of products and development approaches should be followed to increase the portability of the applications.

The recommendations for the software development of Special Applications are the following:

- usage of 4GL whenever possible

- selection of 4GL products among those that are DBMS - independent (possibly complying with the emerging RDA standard to access the database), complying with standards for the User Interface and for the Data Dictionary

- for all other types of applications, usage of ISO ADA, and in cases where integration of COTS software requires the development of "glue" code, ISO C can be used.

- for the design of complex applications and databases CASE tools are used complying with the (emerging) standards recommended in the OSE.

- for database definition and manipulation SQL is used.

5.2.6. Man Machine Interface

The Man-Machine Interface (MMI) handles the interaction between the ADP System and the users most often in the form of a dialogue conducted at a workstation. There are many advantages to the user in having this dialogue conducted in a standard manner, no matter which workstation he is using and no matter which service is being accessed. In addition to being standardised, the MMI should be easy to learn and use, providing a maximum of automation to the dialogue process and a maximum amount of help to use the system fluently without disruption to the user's work.

The current state of the art of system / user dialogue conducted through workstations is the implementation of a Graphical User Interface (GUI) using windowing techniques. The adoption of a standard GUI is recommended for the C^3I system using X. Windows and OSF/Motif for POSIX based workstations and servers and MS-Windows, Windows NT, Windows 95 for DOS based PC Workstations. These recommendations provide for the widest choice of available applications and are similar as to present only minor difficulties to the user if both POSIX platforms and PCs are used in the same system.

5.2.7. System Performance

From the user's point of view the important system attributes which bear on the system performance are:

- speed (response time) determined by capacities and speeds of functional system elements and information processing algorithms,
- system availability determined by the reliability and maintainability of the functional elements,
- Flexibility of the system to accomodate growth and changes of requirements,
- The extent to which stratagic goals including cost reduction measures are incorporated in the system design.

5.2.7.1 Speed

Command and Control involves a decision making process which is built around a requirement to monitor events, assess situations, plan actions, decide on actions and execute and follow up decisions. Much information is gathered, transported, correlated and presented in this process but the decision process is measured in hours and minutes rather than in minutes and seconds, for Command and Control activity at the strategic level.

The performance levels in speed required from a C^3I system, therefore, are not substantially different from those of commercial organisations such as banks. Such performance levels are achievable with commercially available ADP hardware and software.

Speed does, however, remain an important issue at the implementation stage of projects. System design will have a major impact on the performance levels eventually achieved. Selection of hardware and software products, the combination of those products, the efficiency of new software code and data base designs will all play a part in determining the speed performance of the system when fielded.
Whilst the Architecture has provisions such as estimation of traffic load (See Annex 6-A) and storage capacity and resort to hardware/software benchmarking directed specifically at the achievement of given performance levels; its open and modular nature facilitates remedial actions which may become necessary if performance becomes a problem during the implementation stage. Additional hardware may easily be added to the distributed modular structure proposed and off-the-shelf software products can be more easily exchanged if open system standards are followed.

5.2.7.2 Reliability and Maintainability

The techniques for obtaining adequate level of availability are to use fault tolerant or high availability components [5.16, 5.17]. However, the requirements for reliability and maintainability are not expected to be beyond what is normal for commercially available equipment in which rapid and easy replacement of components or modules is the state of the art. The distributed modular architecture also facilitates maintenance with a minimum of disruption to the operation of the system.

5.2.7.3 Availability

In general it is not expected to be necessary to have redundant components in the nodal architecture to meet availability requirements. The one area where redundancy would be advantageous is in the mirroring of data bases and other disk files where possible. This technology is generally available at low cost and offers the ability to work around what is probably the most common hardware failure - head crash on disk - with little inconvenience.

5.2.7.4 Flexibility and Growth

Flexibility is achieved through the modular nature of the Architecture and the provision of CASE tools to regional support centres. Specifically these attributes permit the development of new applications, increases in processing power and storage capacity and the inclusion of new hardware technology provided that it fits within the open system defined by the Reference Model at the time of introduction.

5.3. COMMUNICATIONS NETWORKS

The communications networks that are used in a C^3I System are of three types:

- Loca Area Networks (LAN)
- Metropolitan Area Networks (MAN)
- Wide Area Networks (WAN)

Local Area Networks which serve as subscriber loops with distributed switching ability are very much an integral part of a node system and are interconnected to MAN's or WAN's via gateways and routers. These are described in Sec. 5.2. The following sections will treat the subjects of MAN and WAN and describe the various techniques and technologies with standards that are available or emerging from which a selection can be made for use in a system to suit the needs of the user.

5.3.1. Metropolitan Area Networks

As the services like imaging, video conferencing, multimedia, high resolution graphics etc. have found their application in the market, the need for an interconnectivity having a traffic capacity to handle these facilities over an area to cover the boundaries of metropolitan area of a city or an entire university campus urged for a new network called as Metropolitan Area Network (MAN) [5.4, 5.9]. A MAN can be defined as a network which is capable of providing different types of high speed traffic (such as data, voice, and video simultaneously) across the distances ranging between 5 and 50 kilometers. MAN essentially is a very large dual bus LAN using a Medium Access Control Protocol (MACP) less sensitive to network size than those used in traditional LANs.

The approved MACP by IEEE and ISO is called as Distributed Queue Dual Bus (DQDB) standard. The corresponding standards IEEE 802.6 and DIS 8802.6 specify the following criteria for MANs:

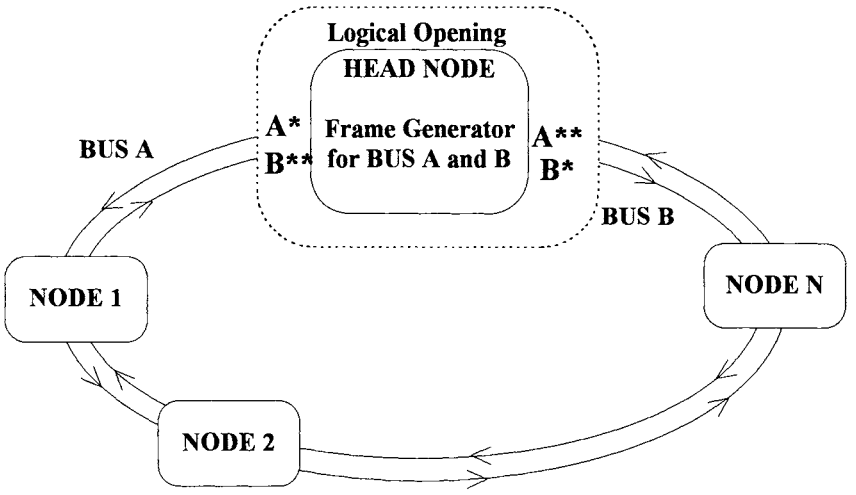
- Provide security that permits the establishment of virtual private networks within MANs,
- Ensure high network reliability, availability, and maintainability,
- Implement a fast and robust signaling scheme,

- Provide efficient performance regardless of size, and
- Support and permit integrated packet switched data and isochronous voice and video services.

The above requirements can also be fulfilled by FDDI which is described in Annex 5-A , specifically FDDI-II version. In the following sections a brief description of DQDB will be given, and comparison of the two alternatives will be made for the final choice for C^3I system network solution.

5.3.1.1 Distributed Queue Dual Bus

The DQDB based MAN involves a dual, counter flowing, unidirectional loop of medium, generally optical fiber, arranged as a physical ring but acts as a dual bus (Figure 5.13). A MAN normally consists of one or more interconnected DQDB subnetworks. The interconnection of MAN DQDB subnetworks could be via bridges, routers, or other LAN interconnection devices in accordance with the DQDB protocol. The MAN normally provides point-to-point communications with station nodes logically adjacent to two unidirectional buses with no end connections. One unit, serving as master (usually the bus head node but not necessarily so), does not repeat data in from either direction. This means that traffic information terminates at the end of the bus and does not cycle around the buses as it does in the ring type topology.



Note: A* Start of information flow of BUS A
A** End of information flow of BUS A
B* Start of information flow of BUS B
B** End of information flow of BUS B

Figure 5.13 A Conceptual View of a MAN With Network Nodes Arranged in a DQDBI Coupled Dual Bus.

The logical bus architecture offers several advantages:

- Network maintenance from a single central office,
- No circulation of packets or alarms since they simply fall off the end,
- High survivability and reliability; if a line segment fails the logical opening of the ring is relocated at the point of failure,
- Efficient slot scheduling for information transport; once a packet is transmitted over a vacant slot, another reservation can be made.

Although a MAN is optimized to cover a perimeter of at least 50 km in radius, its coverage distance and the number of nodes attached to the bus are virtually unlimited. This is largely because all nodes on the bus(es) may be synchronized with the central clocking source externally or internally regardless of their position relative to the clocking source, and also because of the distributed nature of the DQDB, MACP, A MAN station node is an addressable logical and physical attachment to both buses for the purpose of transmitting and receiving information on that subnetwork. External devices can be attached to the MAN through network station nodes.

Similar to FDDI, the DQDB subnetwork component, which includes the MAC and physical layers, is also developed within the traditional LAN technologies and is intended to interface with the IEEE 802.2 standard LLC sublayer. Together, they support and provide OSI physical and data link layer services and functionalities of a MAN subnetwork. Because the MAN technology is designed to support both synchronous video and voice services, and asynchronous data services without bias, its MACP element is different from the FDDI MACP element.

One noticeable distinction is that the DQDB MACP specifies that protocol data unit exchanged between two DQDB MAC entities must be in fixed length. This implies that before the MAC data frame is given to the physical layer for transfer, the variable length MAC data units are segmented into fixed length slots. These slots frame delimit each MAC data unit. When the data units reach their destination (i.e. the peer MAC entity), they are reassembled by the peer MAC entity into their original form. The FDDI (I and II) MAC protocols do not require nor support the process of MAC data unit segmentation and reassembly [5.5, 5.6, 5.7].

The MAN technology with DQDB standard can be applied to following activities:

- Internetworking of high speed LANs including the FDDIs.
- Interconnection of mainframes, peripherals, and private branch exchanges.
- Provision of digital backbones and links to WANs including X.25 and ISDN.

- Provision of integrated services (voice, data, and video) with guaranteed bandwidth allocation.

- Provision of connectionless high speed packet switched data services (i.e. Switched Multi-megabit Data Services -SMDS- provided by public carriers).

- Support of medical imaging and reliable information transfer with multiple security levels. Capable of supporting multiple security labels at different security levels upon information transfer.

- Data transfer between front end high speed CAD / CAM workstations.

- Video and full motion conferencing.

- Allowing a seamless integration with Broadband ISDN (B-ISDN), and as an interim B-ISDN solution.

5.3.1.2 Comparison of FDDI and DQDB As a Backbone Network for a Military C^3I System

It seems to be reasonable that high speed network technology and optical fiber transmission medium are indispensable for advanced integrated communication services. FDDI and DQDB are the high speed networking technologies that are intended to support integrated services such as transparent data, isochronous voice and video information to users scattered around a greater geographical area than it is served by the first generation LANs (ETHERNET, Token Ring, etc.) and both are using optical fiber as transmission medium.

These two networking technologies have some similarities and differences that can be summarized as follows:

- Both technologies offer user data rate higher than 30 Mb/s, and provide integrated services (data, voice, and video).

- Both technologies feature the self healing capability and low Bit Error Rate (BER) which influence the network reliability and survivability.

- FDDI is considered as a LAN with extended network span. DQDB has virtually unlimited network size and span and considered as WAN by many users.

- FDDI architecture usually centers around a single user site while DQDB architecture serves users scattered around multiple sites or around a metropolitan area.

- FDDI networks can be owned and maintained solely by a single user community. DQDB on the other hand, is normally owned and operated by

a public operator or by PTT. In other words, FDDI can be used as a high speed private network while DQDB is an extension of public telecommunications network.

- FDDI technology is more suitable for bursty and continuos type data traffic while DQDB favors the continuos stream of data.

- Because of the inherent characteristics of optical fiber transmission medium, both networks are immune to noise disturbances and probing by the third parties which is very important for a military system.

- FDDI conformant networks are commercially available and implementable. DQDB conformant networks are still in development phase. DQDB will, however, be the basic subsystem compatible with ATM type networks (which may be considered as the third generation LAN).

As it can be seen from the above comparison, FDDI technology has advantages over the DQDB technology for a military system that seeks privacy, survivability, security and expandibility as the main constraints.

5.3.2 Wide Area Networks

While the local and Metropolitan Area Networks are the domains of a group or groups of users with common interest, usually confined to distances of 50 kms or so, Wide Are Networks (WAN's) go beyond this range and cater for the needs of the general public including the long distance communications needs of private user or user groups and are also referred to as public networks. Traffic on these networks have been and still are dominated by telephone and hence the name The Public Switched Telephone Network (PSTN). As is well known, this situation has been changing very rapidly over the last two decades or so as a result of the development of the computer creating needs for various data and imagery services as well as for the integration of these services.

5.3.2.1 Narrowband ISDN

ISDN that has been promulgated by CCITT (ITU) in the period 1981-1988 which uses a limited set of channel types access configurations, protocols and user-network interfaces, has gained international acceptance and as a result nations have been implementing ISDN networks of the narrow-band variety (N-ISDN). These networks can support the communication needs of most C^3I systems which may be implemented in the near and medium term by providing capacity and switching capability ranging from circuit to packet and message switching. These networks are reliable and redundant and therefore can provide survivable links between the nodes of a C^3I system. How these networks are designed and may be used for both civilian and military purposes are described adequately in Ref. [5.3].

5.3.2.2 Broadband Services

Because of the increasing demand by private and public bodies for high-bandwidth applications such as desk-top video conferencing, remote training, there has been a need for broadband services. As far as C^3I systems are concerned, here too, there has been an increasing demand for more PC and WS class terminals with higher speeds to be connected to LAN's which require higher bandwidths. Traditional LANs, such as contention-based Ethernets and token-passing rings, suffer performance degradation when heavily loaded. One way to improve performance and decrease delays is to segment the LAN into multiple subnetworks, interconnecting the smaller LANs with bridges and routers to provide the connectivity of the original network. The result is a larger number of smaller networks. Applications on today's LANs that require high bandwidth include client/server applications, multimedia applications, and graphical user interfaces.

This hunger for bandwidth is translating also to the WAN environment as LAN interconnection becomes increasingly important. The public network has, historically, been the bottleneck in long distance data communication. The Experimental Ethernet operated at 3 Mbps in 1975 and while this might be considered slow today, it was significantly more speed than could be achieved from public network service offerings at the time. By the time T1 (1.544-Mbps) private line services became available in the United States, products providing 10, 50 and 80 Mbps on the premises had already started to appear.

This picture has been changing as public WAN service speeds approach or exceed the capabilities of premises networks with the introduction of such public services as FDDI, SMDS, and ATM. Today's WANs and MANs must offer LAN-like performance- high speed, high throughput, and low delay-over a larger geographic area. Again, it is often the same technology that is demanding the high speeds on the premises that is achieving the high-performance levels in the network.

Market research indicates a general strong trend for a significant, continuing increase in bandwidth demand for advanced communication services. A standardized technological framework for meeting this demand began to take shape twelve years ago in International Telegraph and Telephone Consultative Committee (CCITT)-now known as the International Telecommunications Union Technical Standards Sector (ITU-TSS). That framework called Broadband ISDN (B-ISDN) is essentially complete today. BISDN has the potential to be the next generation infrastructure not only for the telecommunications carrier's public networks, but also-in the context of increasing demands for sophisticated high-bandwidth, or broadband, services- for customers' private network, including local-area, metropolitan-area, and wide-area networks. This remarkable new synergy between premises networks and public networks offers customers dramatic new possibilities of seamless end-to-end information networking. Together with new techniques for transporting voice, data, and video, BISDN makes it possible to deploy multimedia services with powerful signaling and control capabilities.

5.3.2.3 Asynchronous Transfer Mode (ATM)

High-bandwidth multimedia services and applications seem to be best offered by broadband ISDN, based on asynchronous transfer mode (ATM) protocols [5.11, 5.13, 5.14 and 5.15]. These protocols define a structure for the "cells"-data packets of a fixed length, with standard header and payload attributes-that both carry and route information through the network. BISDN is the standard term used to identify high-speed digital networks using ATM as the transfer mode and ITU-TSS recommendations, such as Q.93B, for signalling and call control.

The single most important aspect of BISDN is its integrated support of a wide-range of services and multimedia applications, including any combination of audio, video, and data, not only in the same network but also as part of the same call configuration. A key element of service integration for BISDN is the provision of a wide range of services to a broad variety of users utulizing a limited set of connection types and multipurpose user-to-network interfaces. But BISDN is not limited to connecting customers to high-bandwith digital transport facilities; it also gives users additional capabilities to control and manage the type of connections, the quality of service, and the number of endpoints involved in a call.

BISDN supports switched and permanent, point-to-point and point-to-multipoint connections. These connections in turn support circuit-mode and packet-mode, variable-bit-rate and constant-bit-rate services. The services provided on BISDN can be single-medium or multimedia, connection-oriented or connectionless, and bidirectional or unidirectional in configuration.

The cell-transport concept and specific cell-transfer principles of ATM are responsible for the great flexibility it provides, in terms of both network access and types of service. Two key ATM concepts, referred to as "virtual path" and "virtual channel", enable easy provision of semipermanent connections. ATM also allows dynamic bandwidth allocation, on demand, with a fine degree of granularity.

Independence of the means of transport at the physical layer is another adavantage of BISDN. In its purest conceptual form, BISDN/ATM technology is tightly coupled both to the use of single-mode fiber and to physical-layer transport as defined for the synchronous digital hierarchy (SDH). ATM, SONET, and SDH are mutually compatible and make perfect sense together in the public netqorks. But the BISDN recommendations also enable the use of ATM technology on premises networks such as multimode fiber or twisted-pair local-area networks (LANs). They also allow for a graceful evolution from today's mix of public and private network-specific technologies to what may become a common technology infrastructure for all.

ATM is a scalable technology and can be used for networks that are geographically large or small, have few or many hosts, or operate at low or high speeds and therefore becomes a strong candidate for use in any C^3I system provided that the services required have been satisfactorily standardized [5.13]. Figure 5.14 shows the status of the ATM standards in 1993 many of which have now been completed.

Standard	T1 S1 ETSI	ITU (CCITT)	ATM forum
Physical Layer - D S-3/E3	✓		✓
- SONET/SDH	✓	✓	✓
ATM layer - Initial	✓	✓	✓
- Final	●	●	●
Data services	●	●	●
Voice services	●	●	●
Video services	●	●	●
Circuit services	✓	●	●
PVC Management	●	●	●
Call control			
- simple	●	●	●
- supplemental services	●	●	●
OA & M	●	●	●

✓ Done
● Near completion

Figure 5.14 Global ATM Standards

5.4 DATA AND MESSAGE EXCHANGE SERVICES

Communications between nodes (internodal communications) will be effected by the use of data and message exchange services which are part of the "Common Services" in Layer 3 of the RM and will be outlined below.

5.4.1 Message Handling

The functions of Message Handling System include:

- Interfaces to Wide Area Networks (WAN):
  - the existing military or civilian organizational network through the related interface standard
  - the planned data transfer system through X.400/MHS on top of X.25 and other transport profiles
  - other leased line/PTT communications (e.g. ISDN) through X.400/MHS on top of standard ISO and CCITT communications interfaces.

- Military or corporate central headquarters message handling functions including (see also Section 5.2.5):
    - reception and submission of all formal incoming and outgoing messages in a standard format .
    - logging and archiving of all formal incoming and outgoing messages
    - retention of audit trails of all message transactions and operator activities
    - translation between various formats
    - origination and delivery of informal messages
    - node-system internal reception, temporary storage and distribution of messages
    - system security administration functions
    - system administration, maintenance and control functions
    - gateway control functions
    - message service functions
    - directory services

Figure 5.15 illustrates the military message handling system architecture. Wide Area Network (WAN) interfaces for message handling services and in the future additional interoperability services will be implemented on a WAN gateway component. Depending on the topology and organisational structure of a node-system, the central message handling functions can be distributed on more than one message server being interconnected through X.400/MHS interfaces on LAN or WAN communications.

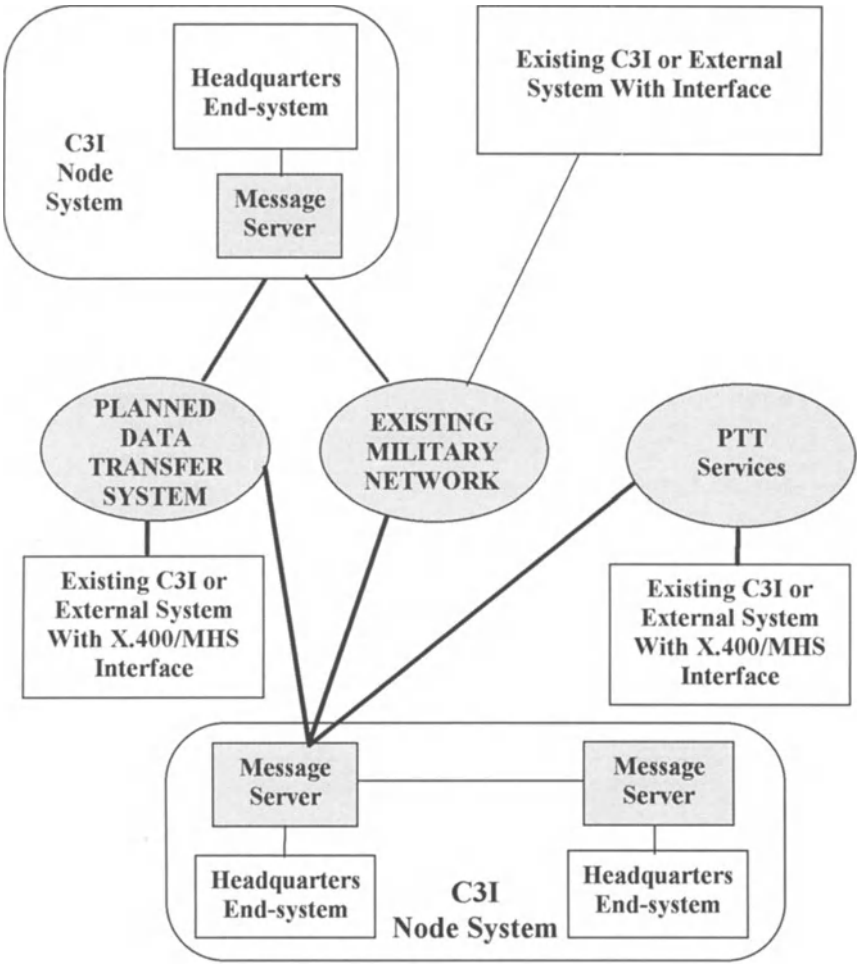


Figure 5.15 Message Handling System Architecture

5.4.2 File Transfer And Access

C^3I system file transfer and access services (at system-wide level) is proposed to be based on the ISO File Transfer, Access and Management (FTAM) standard. FTAM provides a total file handling service among networked systems. FTAM is vendor-independent, i.e. it does not impose any constraints on the local file definitions or implementations.

To achieve independence from vendor implementations, FTAM uses the concept of a virtual filestore, a conceptual, generalised model of filing systems. FTAM allows file handling operations to be specified in terms of operations performed upon the

virtual filestore. At each end system, these operations are then mapped in a local way onto the local operations on real filestores.

The main functions provided by FTAM are:

- transfer; of all types of files to and from a remote node
- access; the manipulation of remotely held data
- management; the remote control of the filestore

FTAM is a powerful, if complex, solution to bulk file transfers and distribute file access over a WAN. It should be noted, that for applications which only require to transfer small simple files, X.400 is sufficient. Some FTAM implementations are known, but FTAM has not yet been widely accepted and implemented in commercial products. The effort to integrate FTAM with COTS applications is considerable. Until FTAM is widely accepted and an integral part in commercial products (e.g. operating system) it is recommended to implement the exchange of data files on the basis of the widely used internet protocol FTP.

5.4.3. Database Access and Replication

Applications may require to access structured data that is stored in another node. The remote access and management of data held in structured databases is best accomplished through a different type of service than those previously described, namely a "remote database access" service. File transfer and message handling would provide a less efficient and reliable mechanism to accomplish this.

For implementing direct access, manipulation and replication of structured data (databases) across different nodes, the following aspects of heterogeneity have to be considered:

- different data models
- different database management systems

The first problem can be solved by standardising the format and semantics of the data subject to potential exchange.

The second aspect is addressed by following a standard protocol for remote database access. ISO RDA (Remote Database Access) is an International Standard defining a protocol allowing a client program to access a remotely located data manager. The so-called RDA "SQL specialisation" deals with the access to a data manager that is a standard SQL database management system (DBMS). The RDA standard permits independence from the specific relational DBMS product accessed.

ISO RDA-compliant products are not yet available. However, products offering RDA-like capabilities are available and an industry standard is expected to be widely supported in the future.

Database replication is a functionality not yet standardised, although RDA provides a basis to implement such a service. Some DBMS provide database replication capabilities, but only for a homogeneous database environment. The recommendations for implementing database replication are the following:

- If the nodes are part of a homogeneous DBMS domain, product-specific replication mechanisms should be used. A COTS solution is expected to be more reliable and cost-effective than any "special" development of such a service.

- If the nodes are heterogeneous (respect to the DBMS), such a service is likely to be developed as a new application. The access to the remote databases should be based on RDA standards to improve the portability.

The compliance with standard data definitions is required in both cases.

5.4.4 Data Interchange Format

Data (e.g. documents, messages, database data) can be transported electronically from one node system to another by means of any of the services previously described, i.e. Message Handling, File Transfer and Access, Database Access and Replication. In order to achieve full interoperability the format and semantics of the data transferred have to be standardised.

The following types of data can be distinguished:

- **Textual messages.** Textual (formal) messages are the objects of the current procedural and operational interoperability standards. The recommendation is to follow them for "formal" messages. No standardisation is necessary for informal messages.

- **Graphics (e.g. situation displays).** International and industry standards are available for graphics data. The recommendation is to use ISO CGM for the exchange of "vector graphics" and (in the lack of an ISO standard) TIFF as the temporary standard for exchange of raster graphics objects.

- **Documents**. Documents can be composed of both text and graphics (e.g. briefing pages). The standard format recommended in the long term for exchanging documents is ISO ODA (and related standards). In the short term, in the absence of ODA compliant products, a pragmatic decision to select an industry standard word-processing product and the corresponding product-dependent format should be made with the widest consensus within organization (at least region-wide, organization-wide).

- **Geographical maps**. No international standard is available for geographical maps. The long term recommendation is to consider the emerging ISO standard (ISO DIGEST). Currently there is no acceptable open standard. Some US and European definitions of geographical products are available within defence organisations (such as the US DMA DCW, Digital Chart of the World) and the various COTS products support one or more of these definitions. In the short term the only pragmatic recommendation is to make a product-dependent decision, with the widest possible consensus (at least region-wide, organization-wide).

- **Structured data**. The format and semantics of structured data exchanged among C^3I system nodes should follow the standard data definitions. The exchange of data with external systems should comply with the applicable, for instance NATO STANAGs on operational information exchange.

5.4.5 Interoperability Services

Interoperability within a C^3I system and between the C^3I system and external systems will be based on:

- **Technical interoperability standards** are based on the adoption of protocols in conformance with the ISO-OSI [5.5] model and corresponding ISO and other international standards. These standards deal with the transport (by electronic means) of data from one node system to another node system.

- **Procedural interoperability standards** are defined (for messages) in terms of message formatting rules and conventions and message set and field format standards. These standards deal, in general, with the format of the data exchanged electronically.

- **Operational interoperability standards** are defined (for messages) in the operational information exchange directives. This kind of directives will be based on message text format standards and syntax, and will be the same system wide directives for operational information exchange. These standards address, in general, the overall (non-ADP) procedure for the exchange of data.

For the Architecture only the first two classes of standards are relevant.

5.4.5.1 System Level Interoperability

Near term interoperability among C^3I node systems could be achieved by means of message exchange (The Message Handling System) because:

- standardization of message handling is well advanced,
- available communications capabilities support message handling services,
- large commercial interest ensures availability of Commercial Off-The-Shelf (COTS) solutions,
- many of the interoperability requirements can be met by message handling:
  - exchange of formal (inter-organisational) messages unformatted or formatted
  - exchange of informal (inter-personal) messages (often referred to as "E-mail")
  - exchange of graphical information by encoding graphical objects
  - exchange of files attached to message bodies; files can contain text, graphics, data-records etc.
  - exchange of multi-media messages, i.e. messages which contain a combination of text, graphics, voice, and possibly video information.

Nodal databases can be updated through data carried in messages. This process is supported through an automated "Message Processing" function. This approach, however is not the most efficient because it requires a man in the loop and the scope of the exchange is limited to defined message text formats.

Providing that common data definitions are established, more direct means for exchange of data between databases can be implemented. For that purpose the database replication service based on the Remote Data Access (RDA) protocol is proposed. This approach will lead to higher consistency between databases in C^3I node systems and more efficient data exchange. When more capable communications means are used, bulk data exchange by means of the File Transfer and Access service, and access of node system databases from remote locations by means of the Remote Database Access service is possible.

5.4.5.2 Interoperability With External Systems

External systems are systems outside the domain of a C^3I system with which the C^3I system requires to exchange information. From military point of view, these include:

- Tactical Information systems (for air, land and naval forces)
- National Intelligence systems
- Other allied and national systems
- Civilian and public systems (e.g. Air Traffic Control system)

In the short term, Message Handling will also be the main data exchange service for implementing interfaces between C^3I system and external systems. Under the assumption that standard message handling interfaces are or will be part of such systems, this approach will avoid costly, special-purpose, bilateral interface developments.

In addition, special-made interfaces with external systems are necessary. In some cases where C^3I system is required to interface with external systems which do not conform to C^3I system architecture, investment in special interfaces on the basis of a case by case decision will be necessary. Special interfaces with external systems dedicated to a single functional area should be implemented on the level of sub-system.

5.5 STANDARDS

5.5.1 Objective

In this section standards will be reviewed for populating the Reference Model given in Chapter 4 necessary for the architectural design.

The dictionary meaning of "Standards" is "Something established by authority, custom or general consent as a model or example". Standards are known by different names depending on the source, for example, standards, specifications, Regulations, Recommendations. By any name, their purpose is to achieve the necessary or desired degree of uniformity in design or operation to permit systems to function benefically for both providers and users. The intended scope of standards can vary. They may be internal within a company or they may apply to an entire country, a world region, or the world as a whole; standards that will be considered here would be given preference in the reverse order, starting with Internationally agreed standards which may be either treaty based or voluntary.

Military and civilian organizations currently face the serious problem of developing an integrated computing environment that takes into account and makes the best use of multiple vendor platforms. For years, various departments inside the same organizations have solved their business problems by purchasing a diversity of hardware platforms, operating systems and applications software suitable for their immediate needs. But these purchases are often incompatible with the solution acquired by a neighbouring department.

While the answers to one department's needs are provided by, for example, a PC-based local area network (LAN) system, another department's problems might be solved by using a high-powered mainframe system. In this case, the hardware and the software of one department cannot be used by the other. Therefore, neither hardware resources nor generated data can be easily shared on-line.

To solve these problems, the system must be designed so that:

- Common hardware and operating system are used as much as possible.

- Application softwares acquired or developed can be used on different platforms (portablity) and can be scaled (scalability) to suit the needs of different users in the organization and these goals are obtained when one subscribes to the principle of open system architecture (OSA) and its standards [5.18].

5.5.2 Developing Applications Software

"Open Systems" have been talked about for some years,and there are many definitions of such systems. The IEEE definition (IEEE 1003.0.01.92) is as follows:

"A system that implements sufficient open specifications for interfaces, services and supporting formats to enable properly engineered application software to:

- Be ported across a wide range of systems with minimal changes.
- Interoperate with other applications on local and remote systems.
- Interact with users in a style which facilitates user portability".

The IEEE defines an open specification as one that is "described in terms of a public specification that is maintained by an open, public consensus process to accommodate new technology over time and which is consistent with international standards".

However, more important than the definition, are the reasons why the Open System approach has been generally adopted. The most obvious are portability, interoperability and distributed processing. These benefits impart the following qualities to open applications:

- **Portability** - Avoids system-specific and non-standard functions in application code so the application can be moved easily between platforms.
- **Interoperability** - Provides applications that can work together in a heterogeneous computing system by sharing data, functions and a common user interface.
- **Distributed processing** - Supports a collection of processes that might use resources (such as physical devices, databases or processing elements) either on a local system or another system on the network.

This, in turn, generally leads to lower cost for hardware and software, lower training overheads and makes it easier to use and maintain a homogeneous multi-vendor environment when based on accepted, in-use, standards. Such standardisation then reduces the overall risks which in turn leads to financial savings, increasing productivity and less overheads to operate systems with differing characteristics.

5.5.3 Selection Criteria for Standards

This sub-section describes the selection criteria considered important when selecting standards for the C^3I system Reference Model (RM) [5.12] described in Chapter 4. In this context, it will also be necessary to bear in mind that the criteria will vary between organisational units - especially with regard to prioritization, but also with regard to the particular relevance of each item.

5.5.3.1 Existing HW and SW (The installed base)

This is important because, even though a military or civilian organisation decides to standardize on a computing model for the coming years, the Organization will have to take account of its existing investment in:

- Hardware
- Software from different vendors
- Self-developed applications
- Know-how (training and experience)

Since a new model will not necessarily be possible to be implemented on an existing equipment, there will be a need for continuity in all running applications due to operational requirements.

Lastly, there will be a need for HW, SW and applications where no standards yet exists. Under these circumstances depending on the urgency of the requirement one can wait till standards become available or one can develop solutions (own or industry provided) tailored to one's own urgent needs.

There will still be areas where the use of proprietary products and services are recommended, but in such cases the organization should strive for a "clean" interface between those systems and systems based on a standard architecture.

5.5.3.2 Security

The security requirements will differ from system to system dependent on the information handled by the system and how the various forms of threat are defined regarding the system.
Military organizations, and more and more with time civilian organizations, have a formal requirement regarding security for their data processing systems.

To meet the formal security requirements the software acquired or self developed must be evaluated/assessed and certified for use at the required security level (See Chapter 7).

The requirement to use certified products, however, limits the set of products and functionality that can be obtained.

In general, security requirements (standards) must be taken into account in the planning, design, development and implementation of the computer hardware and software.

5.5.3.3 Availability of COTS Products

There are a number of reasons why an organization should strive for more use of "Commercial Off-The-Shelf" COTS products. Some obvious reasons would be:

- Better price-performance ratio,
- Less expensive service and maintenance,
- Broad range of products to choose from (HW, SW and standard solutions), means choosing best suitable product for meeting the need,
- Use of latest technology available.

Another reason for selecting COTS where available, is the trend towards the situation that it is the commercial part of the data industry that drive the technology. Earlier this was to a larger extent driven by military applications.

5.5.3.4 Functionality

Does the chosen standard have any effect on the needed functionality? How does for example user interface/user friendliness affect security. Military organizations in general have a need for secure solutions. This may in certain areas affect user friendliness. Another factor is performance, i.e. will a chosen standard give less performance.

5.5.3.5 Portability

To make portability of applications software between many computer systems possible, operating systems and their entire environment must be hardware independent. Portability here is to be understood as the problem-free transfer of software from one system platform to any other system (vendor independent). This is a very important consideration for a heterogeneous environment.

5.5.3.6 Scalability

Open Systems will facilitate scalability of computer systems. The same applications and the same software may be used on all system classes without computer specific restrictions. This include:

- Computer Memory,
- CPU capacity,
- Secondary Storage performance,
- Secondary Storage capacity.

It should be possible to upgrade a system regarding the above mentioned factors incrementally.

5.5.3.7 Interoperability

In an international or nation-wide organization, this selection criterion is one of the most important. C^3I systems require consistency and integrity of data throughout the system as well as interoperability between C^3I applications.

Sharing data and other functions across a network for applications of different types using software obtained from different vendors to run on different platforms is another important selection criterion.

5.5.3.8 Survivability

Survivability is another very important selection criterion. Autonomous operation, in the event of operational failure of parts of the C^3I systems, must be possible and this requires real-time update of data bases made possible by a distributed database management system supporting 2-phase commit protocole.

5.5.4 Recommended Standards

This sub-section recommends standards for each layer of reference model as given in Chapter 4 based upon the criteria established in the previous sections. Figure 5.16 shows in a matrix form the layers and the criteria to be taken into account in selecting standards for the layers and services provided.

Layer in RM / Selection Criteria	Hardware	Operating System	Common Services					Applications	User Interface
			Network	Data Interchange	Data Management	Language	Graphics		
Availability of Standards									
Functionality		Trade-offs will have to be made, both for the organization and functional area wide technical infrastructure							
Interoperability									
Scalability									
Portability									
Survivability									
Security									
Availability of COTS Products									
Installed Base									

Figure 5.16 Selection Criteria and Reference Model Layers

In the following sections, international/regional and industry standards are listed for each layer of the reference model from which standards for populating the reference model layers will be made based on the selection criteria in the previous section.

5.5.4.1 Standards for Hardware Platforms

The hardware products with different functionality in question here range from personal computers, workstations and servers to minicomputers and mainframes

which are available in the market to meet the different requirements of the different users of the system who would have to select the hardware best suited to their needs.

As to the interoperability criterion, the compatibility of hardware platforms with each other necessitates the selection of products which can function together and exchange data without any additional conversion or interfacing requirements.

The next criterion to consider in selecting the suitable hardware platform is scalability, i.e., the capability of the hardware to expand with regard to:

- Memory,
- CPU capacity,
- Secondary storage capacity.

Another relevant criterion for hardware platforms is security. This criterion may require the use of terminals with no non-volatile memory and/or Compartmented-Mode Workstations (CMWs) (See Chapter 7) and electro-magnetic shielding (Tempest).

A summary of the standards for hardware platforms is given in Table 5.1.

Table 5.1 Hardware Platform Standards

Attribute	Source	Standards
Processor Architecture	Intel	i - x 86
I/O Architecture	ANSI	SCSI (Small Computers System Integration) X3.131
	IBM	MCA (Micro Channel Architecture) Specification
	IBM	ISA (Industry Standard Architecture) Specification
	IBM	EISA (Enhanced ISA) Specification

As will be seen, these are all industrial standards (of which there were not mentioned here) with one exception and there are no internally agreed standards which can be recommended generally.

Recommended Standard:

None.

Reason:

The selection of standards for hardware platforms to be used in the system depends upon the functionality required and the availability of COTS products they can support. As an example, if we consider the workstation, depending on the user needs, workstations of different capacity will be used, ranging from high-end workstations with multi-tasking operating system, multi-media capabilities and large storage to low-end personal computers implementing only a subset of the complete functionality.

5.5.4.2 Standards for Operating Systems

There is no international standard for operating systems but there are several de-facto industry standards like MS-DOS, WINDOWS NT/95, USL and OSF/1. Consequently to achieve the required interoperability between platforms it is necessary to standardize one interface between the operating system layer and the other layers in the model. A summary of standards for operating system interfaces is given in Table 5.2.

Table 5.2 Operating System Interface Standards

Attribute	Source	Standards
System Calls	X/Open	X/Open Portability Guide Ver. 3 XPG3/XSI System Calls and Libraries Specification
	X/Open	X/Open Portability Guide Ver. 4 XPG4 System Calls and Libraries
	IEEE	IEEE 1003.1 (posix)
	USL	SVID Volume 1
	ISO	DIS 9945
	NIST	FIPS 151-2 Draft
Utilities	X/Open	X/Open Portability Guide (Ver. 3) XPG3/XPG4
	X/Open	X/Open Portability Guide (Ver. 4) XIS
	IEEE	IEEE 1003.2 (posix)
	USL	SVID Volume 2
	ISO	DIS 9945-2

The portability and interoperability criteria to be applied to the choice of operating system ensures that installation and upgrading of computer systems can be effected easily and economically. Support for Open Systems and standards for Operating Systems is therefore the obvious way to go. This would provide the following cost savings:

- Software development, maintenance and training are for less expensive, and organizations gain access to products and technologies from a broader class of hardware and software vendors.

- Organization investments are protected because systems can effectively support heterogeneous hardware.

- Independent Software Vendors have a single, uniform market for their software with the assurance of compatibility across POSIX-compliant environments.

Recommended standards:

From the Table 5.2 it will be seen that the interface standard to be used should be POSIX (ISO/IEC 9945-2, 1993 or the latest version) for all workstations in the system.

For personal computers the de-facto MS-DOS, WINDOWS NT/95 standards are recommended.

Reason:

There is no need to standardize on the operating system itself, but there is a need for standardizing interfaces of the operating system to the other layers of the RM. In this way, the organization will be able to choose the best platform (operating system and hardware) suitable for the system mission.

5.5.4.3 Standards for Common Services

This layer probably represents the most critical issues with respect to interoperability within a large organization. Standardization of this layer is therefore of particular importance.

The information and data services element includes the services needed to accomplish data access and interchange. This element can be subdivided into the following areas:

- Network services,
- Data definition and access by applications,
- Data repositories and management systems,
- Data interchange formats,
- Programming languages,
- Graphics services.

The above mentioned services and functionalities are mandatory for a C^3I system in order for the users to have access to the necessary data and for exchanging data between various functional areas within the system.

a) Network Services

One great promise of information technology for the 1990s is the introduction of world-wide communications networks that will provide information exchange among heterogeneous computers, communication products and systems. Several solutions emerged during the last decade with more or less backing from the formal standards organizations. The two most prominent are the Open Systems Interconnection (OSI) effort by ISO and the Transfer Control Protocole of the Internet Protocol (TCP/IP) of IETF.

A summary of the standard of various origin for network services is given in Tables 5.3 (a) and (b).

Recommended standards:

TCP/IP and TCP/IP with possibility to transition to ISO standard still to be promulgated.

Reason:

TCP/IP is recommended because of a large number of existing systems that support this protocol now and in the near future.

ISO standard is recommended as a goal because the ISO effort is an example of the extensive resources which organizations and governments are willing to commit to developing international open systems standards. The expanding work that is being done on ISO standards, profile development and testing indicates the accelerating drive, world-wide toward the development of real ISO products.

b) Data Interchange Services

Document data interchange and processing is also known as "text and systems" after the official name of the ISO/IEC subcommittee that does much of the formal standardization in this area. The purpose of standards in this area is to facilitate the interchange of documents by means of data communications systems (networking) or by the exchange of the storage media itself (tapes).

Documents are considered to be items such as memoranda, letters, invoices, forms, and reports which may include pictures and tabular material as well as text. The content elements used within documents may include graphics characters, geometric graphics elements, and raster graphics elements, all potentially within the same document.

Table 5.3(a) Network Services and Standards

Service	Characteristics	Source	Standards
Distributed Processing Services			
	Remote Procedure Calls	OSF	OSF/DCE RPC
	Object Request Brokers	OMG	CORBA/IDL
	Time Services and ASN.1 Protocol Specification	OSF ISO	OSF/DCE Time Services and ISO 8824 ISO 8825
Directory Services			
	Directory Service	CCITT ISO	X.500 ISO 9594-1
	Remote Procedure Calls	OSF	OSF/DCE Cell Directory System
	API Specification	X/Open	X/Open Director Service API (XDS)
	Domain Name Services	IETF	RFC 920
Security Services			
	User Security	OSF	OSF/DCE Security Service

Table 5.3(b) Network Services and Standards

Service	Characteristics	Source	Standards
Communications Services			
	OSI	CCITT ISO	X.200 ISO 7498
	TCP/IP	IETF IETF	RFC 793 RFC 791
	Token Ring	ISO	ISO 8802/5 with IBM Extentions
	StarLan	ISO	ISO 8802/3 (1BASE5)
	Ethernet	ISO	ISO 8802/3 (10BASE2, 10BASE5 & 10BASET)
	Wireless	IEEE	802.11
	Serial	EIA CCITT CCITT CCITT CCITT CCITT	RS232C V.21 bis V.35 V.32 V.42 V.42.bis
	FDDI	ISO ANSI	ISO 9314-2 X3.148
	ISDN	CCITT CCITT ANSI	1.430 Q.931 T1.609
	X.25	CCITT ISO ISO	X.25 ISO 7776 ISO 8208
	Frame Relay	CCITT	Q.921
	SMDS	IEEE	IEEE 802.6
	CSPDN	ANSI/EIA CCITT	RS-232C X.21

	NetBIOS	IBM IETF IETF	NetBIOS Specification RFC 1001/1002 RFC 1006
	SPX/IPX	Novel	SPX/IPX Specification
	LAN Manager	Microsoft	LAN Manager Specification
	Apple Talk	Apple	Apple Talk LAN Protocol Specifications
	SNA	IBM IBM IBM IBM	SNA Reference Manual LU6.2 Reference Manual CPI-C Reference Manual APPN Reference Manual
	OSI Transport Stack	ISO	ISO 7776
	Transport Interface	USL X/Open	TLI Streams Library X/Open XTI
	Link Interface	USL	DLPI
	DOS Interface	Microsoft	NDIS-Network Driver Interface Specifications

Recommended standards:

ODA/ODIF (Office Document Architecture/Office Document Interchange Format)	ISO 8613, ISO 8824
SGML/SDIF (Standard Generalized Mark-up Language/SGML Document Interchange Format)	ISO 8879, ISO 9069
CALS (Computer-aided Acquisition and Logistic Support)	U.S. Department of Defence
EDIFACT	

Reason:

ODA/ODIF is a multipart ISO standard that includes sections on document structures, document profiles, office document interchange format, character content architecture, raster graphics content architecture and geometric graphics

content architecture. The ODA standard references and is also referenced by a number of other ISO standards. It is widely accepted in the market.

SGML and its interchange format SDIF, has gained significant acceptance in specially the technical publishing environment, and is widely accepted in the market.

CALS, from the United States Department of Defence, is a set of user standards defining an SGML application for support of named fuctions of the Department of Defence. This set of standards is also likely to be used by other departments and organizations.

EDIFACT is emerging as the world-wide EDI standard. Use of EDIFACT is being mandated by customs and other European governments as part of the "1992 initiative". It has also been mandated by U.S. customs.

c) Data Management Services

Efforts are underway at the international level (ISO/IEC JTC1/SC 21/WG3) to develop and maintain a reference model for database standardization. The model defines a framework to co-ordinate the development of current and future data management standards for identification of interfaces, positioning of interfaces relative to each other, identification of facilities and processes which support each interface and identification of the binding alternatives associated with each interface.

Database access is critical to the function of numerous applications. Sophisticated models and methods for data organization and access have existed for over thirty years, so that some standards such as database languages are quite mature. Other areas are evolving and standards work is in its early phases.

The purpose of a database language standard is to provide portability of database definition and database application programs among conforming implementations. The Structured Query Language (SQL) is a database language that has been standardized at the international level. The current SQL standard specifies data definition, data manipulation, integrity checking and other associated facilities of the relational data model. In addition, the SQL- standard specifies components that support access control, programming language interface and data administration. The standard provides language facilities for defining application-specific views of the data. Both EWOS (European Workshop for Open Systems), ISO and ANSI are working on SQL-enchancement, like SQL2 and SQL3.

SQL-89 is specified for standalone, single-environment databases. Specifications for access to remote heteregoneous sites are under development in an emerging ISO Remote Database Access (RDA) standard.

A remote database access (RDA) standard may be used to establish a remote connection between a database client, acting on behalf of an application and a database server, interfacing to a process that controls data transfers to and from a

database. The goal is to promote interconnections of database applications among heterogeneous environments. The proposed standard is in two parts, generic RDA for arbitrary database connection and an SQL-specialization for connecting databases conforming to the SQL standard. RDA is a two-way service and protocol specification, not a distributed database specification.

The proposed generic RDA standard provides an RDA service interface and an RDA communication element that exists at both the client and server sites. The generic service does not specify the syntax or semantics of database operations sent from client to server. Instead, the standard assumes the existence of a language specialization that specifies the exact transfer syntax for standard operations. The RDA/SQL specialization complements the generic RDA standard for use when a RDA compliant SQL data manager is present at the server location.

Many SQL vendors begin to have conforming client and server products available before the final adoption of the standard. Vendor consortia such as SQL Access and X/Open have products operational and do demonstrate interoperability among different SQL-servers.

Recommended standards:

SQL	ISO 9075:1989
	ANSI X3.135-89
SQL Access	SQL Access
	X/Open Prelim/91/030
	ISO DIS 9579-1
	ISO DIS 9579-2
SQL2	ANSI
	ISO
RDA	ISO 9579
	X/Open X0191/030

Reason:

We recommend SQL and RDA because these are the de-facto as well as de-jure standards for interfacing a database. The SQL standard will now be extended significantly to SQL2 (and SQL3). Organizations are recommended to follow the standardization work on SQL closely.

SQL Access (published by X/Open) have resulted in a number of submissions to ISO Remote Database Access and ANSI SQL2 committees. More vendors are already offering any SQL Access. For an organization, being "SQL Access compliant" means:

- Vendor independence.
- Application developers can now write a single API instead of database-vendor-unique SQL syntax. This will drastically decrease development costs.
- More applications and tools are available for use as clients.

- Expensive proprietary gateways become standardized, leading to lower support cost.
- Database access meets the needs of the "Open" environment, emphasized in this study.

d) Languages

When recommending a programming language, two major factors affect the choice:

- Suitability for the application
- Availability on multiple platforms.

Not all programming languages are appropriate for every application. For example, the C programming language is widely available, but C is not necessarily suitable for heavy financial applications. COBOL is the common choice for financial applications.

Choosing a standard syntax for the language is as important as choosing the appropriate programming language. Not all compilers are the same. Compilers often contain language extensions to support special features available on one platform only. Although these features may be useful, they may not be available when organizations want to move the application to a new platform.

It is important to understand the cost of using such compiler-specific extensions. One way to avoid extensions - and thereby increase the portability of the application - is to code always in standards such as ANSI or ISO standard.

If the organization for some reasons have to use language extensions that are not available in all implementations, the programmer should be sure to isolate the use of extensions in separate modules or by conditioning the source code. It is a good idea to document the reasons for using the extensions, both in the code and in other design documents.

Recommended standards:

The next table (Table 5.5) lists examples of five programming languages together with the standards that this study recommends:

Table 5.5 Programming Language Standards

Language	Common Name	Relevant Standard
Ada	Ada	ANSI/MIL-STD-1815A-1983
C	ANSI C	ANSI standard X3J11
COBOL	COBOL-85	ANSI X3.9-1978 ISO 1539-1980
FORTRAN	FORTRAN-77	ANSI X3.9-1978 ISO 1539-1980
Pascal	Extended Pascal	ANSI/IEEE 770/X3.160-1989 ISO/IEC 10206:1990(E)

Reason:

These are de-facto and de-jure standards. Fourth Generation Languages, on the other hand, are tightly coupled with the applications software and they are not considered here.

e) Graphics Services

There are many de-facto as well as internationally approved graphics standards that are being used in the industry today. Standards work in the languages, data interchange and user interface areas sometimes overlap with the graphics standardization work. For example, ISO has completed or is working on language bindings to GKS, GKS-3D, and PHIGS among others. Here we will take a closer look at GKS, PHIGS and PEX.

(i) GKS

The Graphical Kernel System (GKS, ISO 7942-1985) is a basic graphics system for applications that produce computer-generated, two-dimensional pictures for on-line graphics or raster graphics output devices. It supports operator input and interaction by supplying basic functions for graphical input and picture segmentation. It provides the programmer with the ability to create graphics output on a wide variety of graphical devices. These include a number of existing graphical devices installed throughout an organization. These are devices such as black and white and colour displays, printers, plotters and camera systems. There are a variety of input-devices.

In order to allow particular applications to choose a graphics package with the appropriate capabilities, GKS was defined with different levels. A three-level

structure is defined for both output and input. Most implementations provide all three output and two input levels. The highest input level is not often supported becaues it requires asynchronous input facilities not found in all operating systems.

GKS provide the application programmer with a suite of functions for the management of graphics images. It provides an environment for controlling display and input devices as well as for storing graphics data and recording the progress of a particular processing session.

The standard is language independent, but the programmer must customise an application using this standard for input and output devices.

The Graphical Kernel System for Three Dimensions (GKS-3D), ISO 8805-1988 standard, provides a set of functions for definition and display of 2D and 3D graphical data, storage and manipulation of graphical data and input of graphically related data. The standard is a set of extensions of ISO 7942 (GKS-2D).

(ii) PHIGS

The Programmers Hierarchical Interactive Graphics Standard (PHIGS, ISO 9592) is a functional specification of the interface between an application and its graphics support system.

PHIGS controls the definition, modification and display of hierarchical graphics data. In addition, it specifies functional descriptions of systems capabilities including the definition of internal data structures, editing capabilities, display operations and device control functions. PHIGS was designed to be used in application programs with the following needs:

- A high degree of interaction
- Definition and display of 3D as well as 2D primitives
- Multilevel/hierarchical structuring of data
- Rapid modification of graphics data and the relationships among the data
- Geometric articulation

Objects are defined in the PHIGS graphical database by a sequence of elements including output primitives, attributes, transformations and innovations to other objects, and object pair definitions. These elements are grouped into entities called structures. Structures may be related in a number of ways including geometrically, hierarchically or according to inherent properties or characteristics defined by the application.

PHIGS permits rapid, dynamic access to a centralized graphics database. This allows PHIGS to support interactive application programs, and with the appropriate platform, real-time definition and modification of graphics data.

(iii) PEX

The X-Window System protocol (see User Interface Standards) has a limited level of graphics functionality. But existing standards such as PHIGS and GKS are much more comprehensive. PEX (PHIGS Extension) is an extension of the X-Window System to support 3D graphics through PHIGS.

Recommended standards:

GKS	ISO 7942-1985
GKS-3D	ISO 8805-1988
PHIGS	ISO 9592
PEX	de-facto standard

Reason:

They are de-facto and de-jure standards.

5.5.4.4 Standards for Applications Software

The selected model should define an application software integration model, which is a visual representation of an application software and of its interactions with its environment.

An application software must interact with the following elements:

Users

An application software must display information to, and accept requests from the user.

Data

An application software must read and write data and access information and resources organization-wide.

Other applications

An application software must communicate with other applications software.

Underlying system

An application software must obtain system resources.

The application software has, in essence, a dialogue with each of these elements as shown in Figure 5.17.

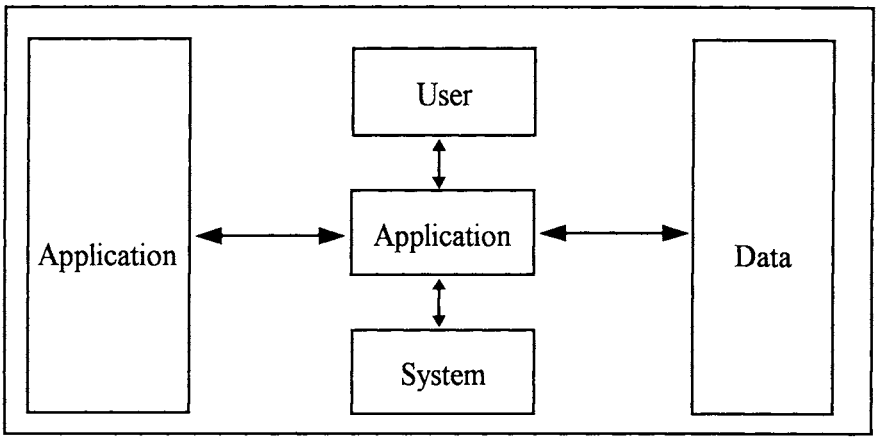


Figure 5.17 Application Software Integration Model

Recommended standards:

Standards for applications software are not recommended.

Reason:

The applications software must be selected by the need of an organization for Office/CCIS/MIS applications. Thus all applications software should be produced by rules and specifications defined by the Open System Architecture principles.

The most important selection criterion for an application software is that it must use the common services supplied by the corresponding layer which has been populated by the standards recommended as in the previous section.

There are four other important selection criteria for selecting applications software:

- Functionality,
- Interoperability,
- Portability,
- Installed base.

5.5.4.5 Standards for User Interfaces

It is essential for an organization that the user interfaces are standardized. This is useful for all organizational activities in large organizations with a heterogeneous environment. An organization needs a user interface that makes workstations, PC's X-terminals and the different programs look the same on all displays. The graphics symbols on all the workstations should be consistent in appearance.

In other words, all the applications software, regardless of the system specific features, should have the same "look and feel".

The best solution for achieving this, is OSF/Motif. OSF/Motif is based on the X.11 standard of UNIX X Windows system, and is written in C programming language. This user interface has already been installed on a wide range of hardware systems of different manufacturers, thus supporting the philosophy of open systems architecture which is really the "backbone" of the design process described in the book.

OSF/Motif is a standard graphics user interface using mouse and windowing techniques. OSF/Motif visually appealing three-dimensional representation with its graphics symbols and pop-up and pop-down menus facilitates operations for interaction with the programs. The graphics design of OSF/Motif is of high quality, and can be adapted simply to meet the individual needs of the user.

Many Organizations have many installed PC's. The amount of PC's will probably, for the years to come, be even increasing. OSF/Motif's "look and feel" is practically identical to MS-Windows running under MS-DOS, and Presentation Manager running under OS/2. This represents a major step forward for all PC users who wish to switch to high-performance graphics workstations or to multi-user systems, or for those users wishing to run PC's, workstations and multi-user systems together in a mixed configuration.

OSF/Motif supports the programmer with a variety of features.

The most important features are:

- The Tool Kit
- The Window Manager
- The OSF/Motif Style guide

All these features enable an organization to program organization-wide applications software using OSF/Motif as a standard user interface understood and recognized by all individual users and user groups.

OSF/Motif is also equipped for use on international markets. In an international organization, Native Language Support (NLS) which enables applications to be adapted simply to the respective national language, is supported in full by OSF/Motif. With its NLS, OSF/Motif meets all the requirements of the X/Open Portability Guide 3 (XPG3), and is therefore currently the market leader from the point of conformance to standards.

OSF/Motif satisfies all the objectives of the open systems architecture objectives striven for by international standard bodies such as the OSF or X/Open fully, and should be the natural choice for the User Interface Layer.

A summary of User Interface Layer Standards for windowing and virtual terminal services is given in Table 5.6.

Table 5.6 User Interface Layer Standards

Service	Characteristics	Source	Standards
Windowing Services			
	Windows Windows NT	Microsoft Microsoft	Windows 3.X Specifications
	X Windows	ANSI OSF	X.11 OSF Motif
Virtual Terminal Services			
	VTS	ISO ISO	ISO 9040 ISO 9041

Recommended standards:

OSF/Motif
MS-Windows (NT)
VTS ISO 9040,9041
Reason:

They are the standards delivered by most vendors.

5.5.5 Recommended Standards Summary

Summary of Recommended Standards for all the RM layers is given in Figure 5.18.

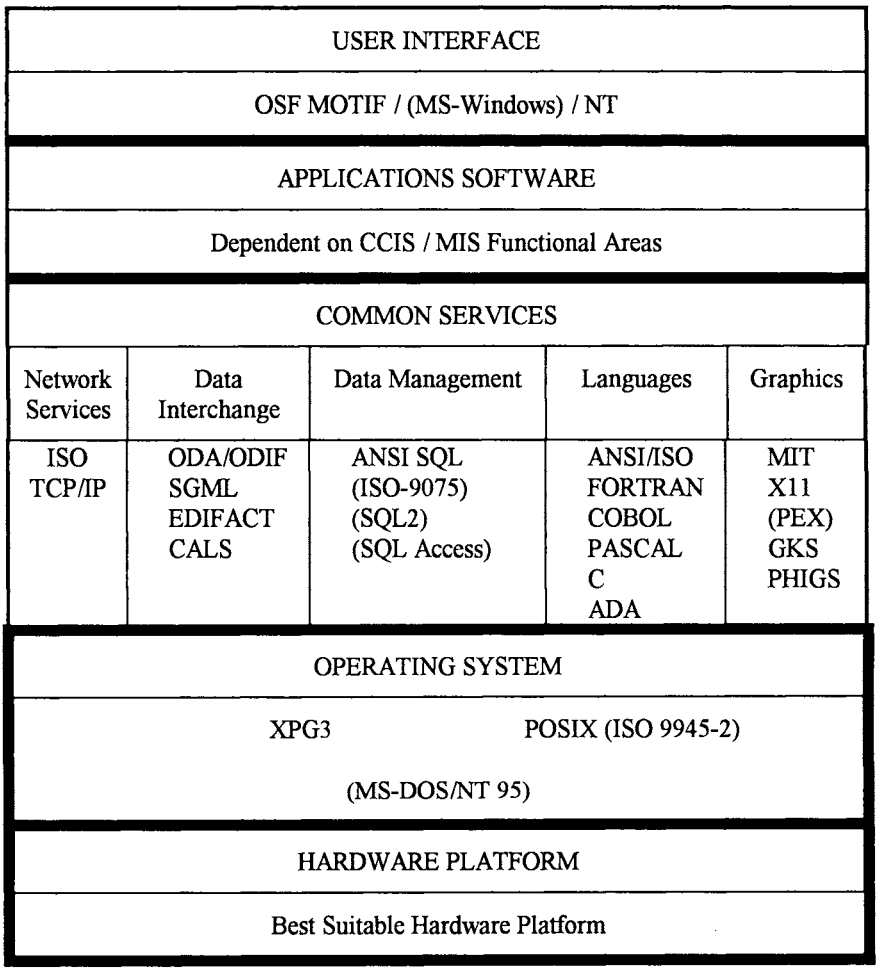


Figure 5.18 Summary of Recommended Standards

A listing of the standards used by the European Commission in their "Guidelines for an Informatics Architecture" is given reference in Annex 5-C which shows great similarity with the standards recomended in this book.

5.6 REFERENCES

[5.1] "C^3I Methodologies, Modelling and Program Management", AFCEA Turkey Symposium, Ankara, 1993.

[5.2] Davidson, R., F., Muller, N. J., "Interneworking LANs: Operation, Design and Management", Artech House Inc., Boston, 1992.

[5.3] Ince, A. N., et al, "Planning and Architectural Design of ISDN", Kluwer Academic Publishers, Boston, 1995.

[5.4] Kessler, G. G., Southwick, P. V., "ISDN: Concepts, Facilities and Services", McGraw Hill, 1997.

[5.5] "Shedding Some Light on Dark Fibre", Data Communications, January, 1994.

[5.6] Ungar, S., "Fibre Optics Theory and Applications", John Wiley & Sons Inc., 1990.

[5.7] Minchandani, S., Khanna, R., "FDDI Technology and Applications", John Wiley & Sons Inc., 1993.

[5.8] Santamarie, A., Lopez-Hernandez, F. J., "Wireless LAN Systems", Artech House, 1994.

[5.9] Held, G., "Internetworking LAN and WANs", John Wiley & Sons Inc., 1993.

[5.10] Cauchi, P., Dennison, S., "Steps for Implementing Local Area Network", John Wiley & Sons Inc., 1993.

[5.11] Patrovina, A., "Non-blocking Architecutres for ATM Switching", IEEE Communications Magazine, February, 1993.

[5.12] International Organization for Standardization (ISO), ISO 7498: Information Processing Systems-Open System Interconnection-Basic Reference Model, 1984.

[5.13] "Broadband ISDN", AT&T Technical Journal, Nov/Dec. 1993, Vol. 73, No: 6.

[5.14] Le Boudec, J. Y., "The Asynchronous Transfer Mode: A Tutorial Computer Networks and ISDN", North-Holland Publishing Company, 1992.

[5.15] Onvural, Raif O., "The Asynchronous Transfer Mode Networks", Artech House, 1995.

[5.16] Flaviu Cristian et al. "Fault-Tolerance in Air Traffic Control Systems, ACM Transaction on Computer Systems", Vol. 14, No. 3, August 1996, pp. 265-286.

[5.17] Gray, J., "Why do Computers Stop and What Can be Done About it?", In the 5th Sysposium on Reliability in Distributed Software and Database Systems, 1996

[5.18] Jain, B. N., Agawala, A. K.,"Open Systems Interconnection", Mc Graw Hill, Inc., 1993

ANNEX 5-A

LOCAL AREA NETWORK (LAN) TECHNOLOGIES AND STANDARDS

1. TOPOLOGIES

The topology of a LAN means the structure or geometric layout of the cable used to interconnect workstations on the network. LANs have flat topologies for peer-to-peer communication. There are three basic LAN topologies in common use-namely bus, ring and star [5.3].

A bus network is a single line of cable to which the nodes are connected directly by taps. It is normally employed with distributed control, but it can also be based on central control. It is a passive network, which means that the signals are not regenerated and transmitted at each node. This topology is mostly, applied for ETHERNET type LANs.

In a ring network, the nodes are linked into a continuos circle on a common cable and signals are passed unidirectionally around the circle from node to node, with signal regeneration at each node. A ring with a central control is known as a loop. Typically, the access method employed in a ring topology requires data to circulate around the ring, with a special set of rules governing when each station connected to the network can transmit data. This topology finds its application in Token Ring, ARC net and FDDI type LANs.

In a star network., each station is connected to the network controller. Then the access from one station on the network to another station can be accomplished through the network controller. The network controller, here, can be viewed as functioning similar to a telephone switchboard, since access from one station to another station can occur only through the central device. The type of LAN utilizes this kind of topology is star LAN and ATM Local Networks.

There are some variations of these three configurations available such as, star-shaped ring, dual ring, star-shaped dual ring, etc. which find recent applications in ETHERNET, Token Ring and FDDI type LANs respectively.

2. ACCESS METHODS

If the topology of a LAN can be compared to a data highway, the access method might be viewed as the set of rules that enable data from one workstation to successfully reach its destination via the data highway. Without such rules, it is quite possible for two messages sent to the same or different address by two

workstations to collide, with the result that neither message reaches its destination. There are three common ways of ensuring that nodes gain orderly access to the network and that no more than one node at a time gains transmit control of the LAN channel. These are namely [5.2]:

- Carrier Sense Multiple Access/Collision Detection (CSMA/CD)
- Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA), and
- Token Passing.

Each of these access methods is uniquely structured to address the previously mentioned collision and data destination problems. The first two methods can be categorized as contention network and the last one as deterministic.

(a) Carrier Sense Multiple Access/Collision Detection (CSMA/CD)

Carrier Sense Multiple Access/Collision Detection (CSMA/CD) is one of the earliest developed access techniques and is the technique used by ETHERNET type LANs that utilize bus topology.

Under the CSMA/CD concept, stations access the network by listening to the network to determine if it is idle (listen-before-talk). Upon sensing that no traffic is currently on the line, the station is free to transmit. If the line is busy, the station waits until it becomes idle prior to transmit data. Since it is possible for two stations to listen at the same time and discover the line is idle, it will then be possible that the two stations could transmit at the same time. When this situation arises, a collision results forcing the stations to back off and try again at staggered intervals. The more stations that are connected to the network mean the higher the probability that such collisions will occur. The access method CSMA/CD regulates how two or more stations share the common bus transmission medium. Typically, each station will use either a randomly generated or predetermined time-out period prior to attempting to re-transmit the message that previously collided.

The CSMA/CD access technique is best suited for networks with intermittent transmission, since an increase in traffic volume causes a corresponding increase in the probability of the line being occupied when a station wishes to transmit. However, there is a practical limit on the length of bus networks. The corresponding IEEE standards 802.3 recognizes 2500 meters as the maximum bus length regardless of number of repeaters, speed or cable type. A single cable segment may reach 500 meters before repeaters are used to extend the bus beyond this limit. While such devices amplify and reconstitute weak signals that may cause corrupt data packets, they do not compensate for signal propagation delays that can increase the likelihood of collisions. Therefore the use of repeaters on high-speed LANs does not guarantee improved LAN performance; performance may actually deteriorate with increase in traffic.

(b) Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)

Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) represents a modified version of the CSMA/CD access method. Under the CSMA/CA access

technique, each of the hardware devices attached to the talkers on the network estimates when a collision is likely to occur and avoids transmission during those times. Since this technique eliminates the requirement for collision-detection hardware the cost of hardware to implement this access technique is usually less than that of CSMA/CD hardware

(c) Token Passing

In a token passing access method, each time the network is turned on, a token is generated. Consisting of a unique bit pattern, the token travels the length of the network, either around a ring or along the length of a bus. When a station on the network has data to transmit it must first seize a free token. The token is then transformed to indicate it is in use and information is added which represents data being transmitted from one station to another. During the time the token is in use the other stations on the network remain idle, eliminating the possibility of collisions occurring. Once the transmission is completed the token is converted back into its original form by the workstation that transmitted the frame and becomes available for use by the next station on the network.

3. TRANSMISSION MEDIA

Transmission medium used in a LAN can range in scope from "twisted-pair" wire, as used in conventional telephone lines, to coaxial cable, fiber optic cable and atmosphere used for radio frequency communications. Initially, thick coaxial cables were used for LANs which were inflexible and difficult to install. Later, thin coaxial cabling was developed that was much easier to install. Today, it is common to find LANs operating over both shielded and unshielded twisted-pair wiring. Fiber optic cables which are used to interconnect LANs between floors or between buildings offer total immunity against Electromagnetic Interference (EMI) and Radio Frequency Interference (RFI). Wireless LANs use radio waves to connect stations on the same floor, but then EMI and RFI could compromise security. The use of spread spectrum technologies helps, with further enhancements through the use of encryption algorithms. Table 5-A.1 shows the basic applications and the type of cable used for realization of those applications.

Table 5-A.1 Recommended LAN Cabling

Application	Unshielded Twisted Pair (UTP)	Shielded Twisted Pair (STP)	Coaxial Cable	Fiber Optic Cable
Dial Data	x			
Data / Voice to 64 Kbit/s	x	x	x	x
ETHERNET, Token Ring 4 Mbit/s		x	x	x
Token Ring 16 Mbit/s		x	x	x
FDDI (Fiber Distributed Data Interface)				x
DQDB (Distributed Queue Dual Bus)				x

4. TYPES OF LANs

In the Section above three main types of LANs were named (ETHERNET, Token Ring, and FDDI) in the discussion of topology of LANs. Apart from those LANs, several different types of LANs have found application, either vendor dependent or non proprietary types, during the evolution of computers and communications networks since from 1970s to 1990s. In this book, LANs will be classified according to the evolution phases and the term "generation" will be used in the classification [5.2].

4.1 First generation LANs

First generation LANs were developed in 1970s to provide high speed local connectivity among user devices. A distributed medium sharing discipline was employed. The ETHERNET technology was then brought into the scene by a joint effort among Xerox, Intel, Digital Equipment Corporation (DEC). ETHERNET initially employed coaxial cable arranged in a logical bus operating at 10 Mb/s. For the time being however, unshielded and shielded twisted pair cables are also widely used. Its usage area is usually limited to a section of a building or the entire building. Because of the CSMA/CD protocol, ETHERNET is not advisable for campus type or metropolitan area applications.

Another first generation LAN technology used is called Token Ring which has a topology of a closed looped ring and operates at rates of 4 Mb/s or 16 Mb/s. A set of bytes called "token" are circulated round the ring giving each station in sequence a chance to put information on the network as described above. Because each node

acts as a repeater such that, data packets and the token are regenerated at their original signal strength, such networks can be applied for longer distances such as university campus areas or in factories. Another advantage of token ring network is that high-priority traffic takes precedence over lower priority traffic. The main disadvantage of token ring configuration however, is that the failed nodes and links can break the ring preventing all the other terminals from using the network. A dual ring configuration with redundant hardware and bypass circuitry provides isolation of faulty nodes from the rest of the network, thereby increasing reliability.

At first, there were no common standards for LANs, and different companies utilized different approaches. Extensive standardization efforts have been spent by Institute of Electrical and Electronics Engineers (IEEE) over a long period, which resulted the well-known IEEE standards extensively used for first generation LANs such as IEEE 802.2, 802.3, 802.4, and 802.5. International Standards Organization (ISO) has also issued corresponding standards such as ISO/IEC 8802.2, 8802.3, 8802.4 and 8802.5 on the same subject.

4.2 Second Generation LANs

The capabilities of all the first generation LANs are limited in distance and bit rate, because these LANs are basically designed to interconnect data processing equipment (host computers, personal computers, plotters, printers, servers, etc.) in one building or in one floor. Its physical span is therefore limited to a few kilometers, and data rate reaches typically to 10 Mb/s for ETHERNET type LANs and 16 Mb/s for token ring LANs.

With the large base of independently installed LANs a new requirement has emerged. The developments in technology and the increased needs to expand the facilities forced a bigger capacity LAN that can be established in a broader region. Some of the reasons, which led organizations and industry search for a more capable system to meet the increasing requirements, can be summarized as follows:

- Increasing number of users being added to the network,
- Enhanced computing power of smaller desktop system and growing requirements for networks to support graphic applications,
- Development of client / server facilities,
- Increasing traffic loads on existing backbone networks,
- Possibility of deployment of a new, high speed applications being realizable,
- The need to interconnect various LANs in a regional area like a university campus or a large organization.

A system to cover the above requirements over a limited distance with a large data rate came into the scene as another generation of LAN which was defined as High Speed Local Area Network (HSLAN). In this book however, the term "Second Generation LANs" will be used for HSLANs. A standard application of second generation LAN is Fiber Distributed Data Interface (FDDI) which will be described in detail in the following section.

5. FIBER DISTRIBUTED DATA INTERFACE

The Fiber Distributed Data Interface (FDDI) is a set of standards that define a 100 Mb/s shared medium LAN utilizing single mode and/or multimode fiber and twisted pair cabling. An FDDI ring network normally consists of a set of stations that are logically connected in serial and a transmission medium to form a closed loop. Each station in FDDI ring is an addressable logical and physical attachment capable of transmitting, receiving and repeating information [5.5, 5.6, 5.7].

Three types of FDDI stations are defined according to their attachment types, and the number of Medium Access Controllers (MACs) which support the FDDI Medium Access Control Protocol (MACP). These station types are:

(a) **Dual Attachment Stations (DASs).** This type of FDDI station has two physical attachments, one to each dual ring and one or two MACLs. High end workstations, servers, mainframes, bridges, and routers are generally DASs. DASs are also referred to as Class A stations.

(b) **Single Access Stations (SASs).** Single access station has only one physical attachment to either one of the two rings and only one MACL. SAS attaches to one of the dual rings through a concentrator station. Low end workstations, and personal computers are the examples of SASs. This type of station is also called as Class B station.

(c) **Concentrator Stations.** An FDDI concentrator station is a special one that has attachments to the ring and multiple ports to facilitate the connections of SASs to the FDDI network. There are two types of concentrator stations, namely dual attachment concentrators (DACs) and single attachment concentrator (SACs). A DAC has two physical attachments, one to each of the dual rings. A SAC has a single physical attachment attaching it to a concentrator only.

The concept of an FDDI network with the variety of above defined stations in use can be established as shown in Figure 5-A.1 The primary ring in the figure is the active ring which carries traffic among the stations at normal operating period. The secondary ring is the backup or standby ring that is normally idle and carries traffic when there is a service disruption or reconfiguration of the primary ring as a result of a station failure or a break at the fiber optic transmission medium. This sort of arrangement provides the system a built-in resilience and survivability.

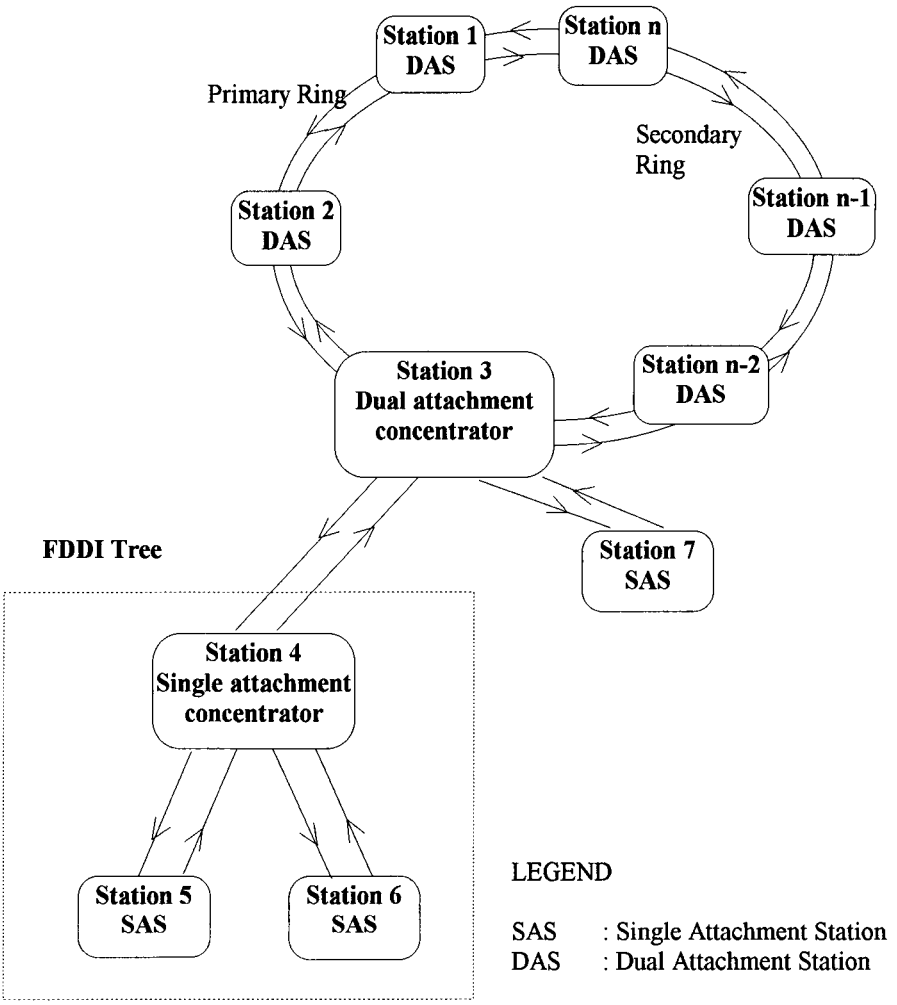


Figure 5-A.1 A Conceptual view of an FDDI network with the variety of station types

5.1 Related Standards

The protocol architecture of the subnetwork component of an FDDI network is developed closely with the traditional LAN architecture within the OSI layering framework. In this connection. IEEE 802 standard defines a layered structure for the two lowest layers (physical and data link layers) of the OSI model. The relationship between IEEE 802 and OSI model is established as shown in Figure 5-A.2. Three of the four layers defined are LAN dependent while the fourth one - logical link sublayer - independent of the LAN technology used as specified in IEEE 802.2.

FDDI type LAN technology is specified in a set of standards proposed by ANSI Accredited Standards Committee (ASC) and approved and published by ISO and ANSI. The sets of these standards are composed of four parts covering Physical Medium Dependent (PMD), Physical Protocol (PHY), Medium Access Control (MAC) sublayers, and Station Management (SMT) plane which is shown in Figure 5-A.2. There is no specific LLC sublayer standard for FDDI, because FDDI is specified to be compatible with LLC sublayer.

There are two version of FDDI, namely FDDI-I and FDDI-II. FDDI-I provides both synchronous and asynchronous packet-switched data services. FDDI-II on the other hand, supports both the services supplied by FDDI-I, and provides also isochronous circuit switched voice and video communications services. For FDDI-II specific applications, MAC protocol for FDDI-I is developed to meet FDDI-II requirements. MAC protocol for FDDI-II is defined as Hybrid Ring Control (HRC) protocol. The relationship between FDDI-II standards and OSI model is shown in Figure 5-A.2.

As for the PMD standard, two versions are defined depending on the medium:

- The basic PMD with multimode fiber,
- SMF-PMD with single mode fiber.

The first version operates at 1325 nm optical window and light emitting diodes (LEDs) are utilized. This version is selected for short distances and cheap solutions whereas the second version is utilizing single mode fiber and laser transmitter and applied for long range solutions up to 100 Km. A combination of two versions are possible on a single FDDI ring, by applying the first version for short distances and second version for the main link and long distances.

5.2 FDDI Applications

FDDI has found a widespread application range due to its capability of comparatively larger data rate than the predefined first generation LANs, and the possibility of coverage of broader regions as a single LAN, and even acting as a Metropolitan Area Network (MAN). Some of the potential application areas of FDDI technology are as follows:

- As a backbone network for first generation LANs (ETHERNET as per IEEE 802.3, token ring as per IEEE 802.5 etc.)
- As a HSLAN for the activities like processor to processor communications, distributed processing, high speed peripheral communications, file transfers, graphics, etc.
- As gateway network connecting high performance workstations and supercomputer,
- Support of integrated services and provision of compatible ISDN B channel services (specific to FDDI-II),
- Provision of secure isochronous communication channels (specific to FDDI-II).

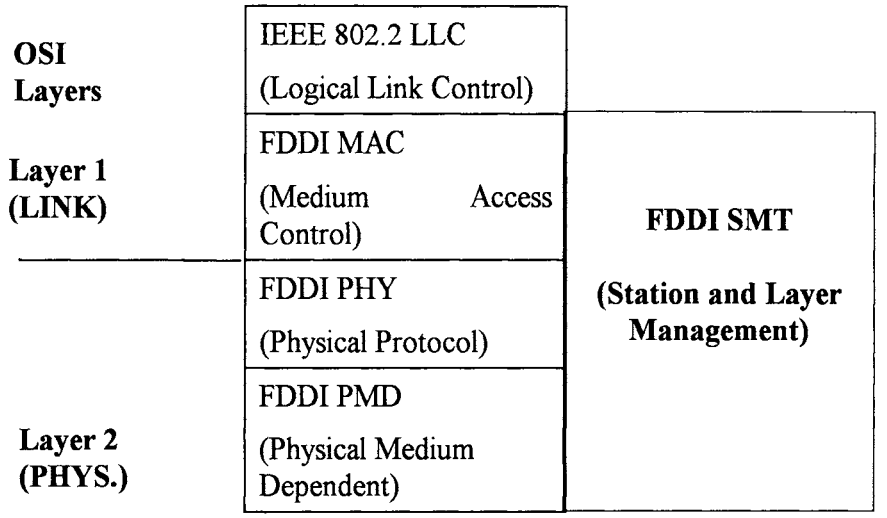


Figure 5-A.2 FDDI-I Standard and its relationship to the OSI layers

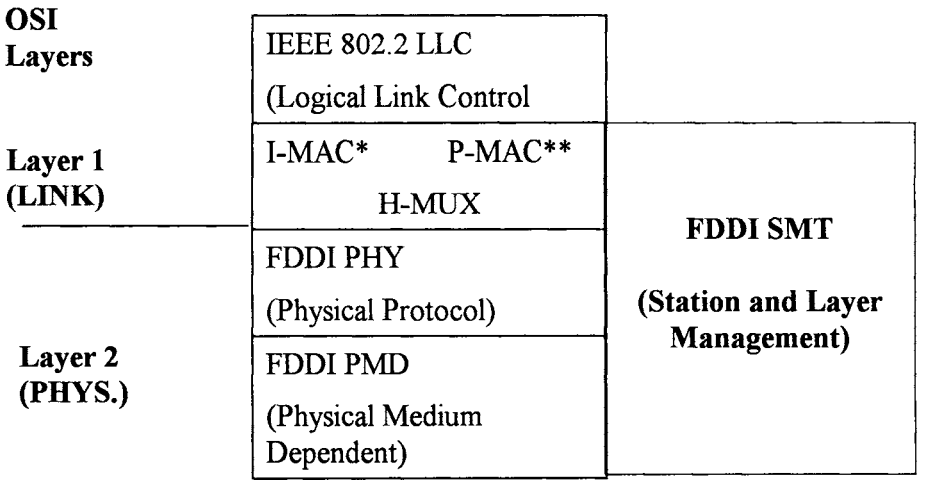


Figure 5-A.3 FDDI-II Standard and its relationship to the OSI Layers

* Isochronous Voice Medium Access Control
** Packet Data Medium Access Control

ANNEX 5-B

ABBREVIATIONS

AES	OSF Application Environment Specification
ANSI	American National Standards Institute, U.S. member of ISO
API	Application Programming Interface. Language and subroutine calls
CALS	Computer Aided Acquisition and Logistics Support. U.S. Department of Defence specifications for electronic exchange.
CEN	Comite Europeen de Normalisation (standards). Member countries are EC and EFTA
CCITT	International Telegraph & Telephone Consultative Committee (of International Telecommunication Union - Telecom agency of United Nations)
CMIP	Common Management Information Protocol. ISO protocol used to exchange network management information.
COTS	Commercial Off The Shelf. Means commercial products.
DCE	Distributed Computing Environment. OSF specifications and related technology.
DIS	ISO Draft International Standard (DP accepted, second technical ballot)
DP	ISO Draft Proposed Standard (has started first technical ballot)
EDI	Electronic Data Interchange format (X12)
EDIFACT	A United Nations sponsored group developing international EDI-standards
EN	European Norm (standard form CEN/CENELEC)
ENV	European pre-standard form CEN/CENELEC; revise within 3 years.
EWOS	European Workshop for Open Systems (OSI Functional Standards)
FDDI	Fiber Distributed Data Interface. ANSI standard for high-speed optical Fiber networks
FIPS	Federal Information Processing Standard (U.S. Government)
FTAM	ISO standard File Transfer/Access and Management. OSI file service.
GOSIP	Government OSI Profile (U.S. FIPS 146, U.K., EC version)
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers. Professional organisation
IP	Internet Protocol. Protocol for connectionless-mode services in Internet suite
ISO	International Organization for standardisation
IT	Information Technology. Common term in international sphere.
Internet	Network of many networks running Internet suite of protocols
LAN	Local Area Network (such as ISO/IEEE 802.3...)

MHS	Message Handling System (X.400 name for mail service)
NBS	National Bureau of Standards (now called NIST)
NIST	National Institute of Standards and Technology (formely NBS)
ODA	Office Document Architecture, ISO Stanard 8613
ODIF	Office Document Interchange Format, ISO Standard (part of ODA)
OSF	Open Software Foundation. Consortium developing AES, OSF/1 and tests
OSF/1	First release of OSF's operating system implementation
OSF/Motif	OSF's graphical user interface environment
OSI	Open System Interconnection (communication protocols). ISO 7498-1984
POSIX	Portable Operating System Interface. This interface specification is based on the IEEE 1003.x standard. Suite of API standard.
SQL	Structured Query Language, ISO/ANSI Standard X.3135-1989. Relational Data Base Management System API
SDIF	Standard Document Interchange Format (Interchange format for SGML=
SGML	Standard Generalised Mark-up Language (page formatting)
STANAG	NATO Standardisation Agreement
TCOS	IEEE/CS Technical Committee on Operating System (POSIX sponsor)
TCP/IP	Transfer Control Protocol, for the Internet Protocol: U.S. Department of Defence network.
UNIX	Trademark name for AT&T operating system product
WAN	Wide Area Network
X-11	X Window System data stream encoding (OSI application level), X3H3.6 Standard
X.21	Synchronous bit oriented protocol (OSI layer 1)
X.25	ISO/CCITT standard WAN (long distance/synchronous) protocol (OSI layer 3)
X.400	ISO/CCITT standard mail transfer (OSI layer 7 - application protocol)
X.500	CCITT standard for directory services
X/Open	An international vendor consortium producing the Common Application Environment (XPG3)
XPG3	X/Open Portability Guide 3, contains Common Application Environment API specifications.
X Window	Window system developed by MIt is Project Athena

ANNEX 5-C

STANDARDS USED BY THE EUROPEAN COMMISSION

1. EQUIPMENT

1.1 Connectivity (layer 1-4)

OSI	Open System Interconnection	ISO 7498-1..4, ADD1
PSDN	Packet Switching Data Network based on:	CCITT R X.25
	- T/31 Permanent Access to PSDN	ENV 41104
	- T/32 Switched Access to PSDN	ENV 41105
PAD	Packet Assembly Disassembly: based on CCITT R X.3, X28, X29	ENV 41901
ISDN	Integrated Service Data Network based on:	
	- Plug (socket)	ENV 41104, EN 28877
	- Reference Configuration	ENV 41104
	- Architectural Model, Supplementary Services	ENV 41105
HDLC	High Data Link Control	ISO 3309, 4335, 7809, 7478, 7776
X21 bis	DTE/DCE Interface Circuits	CCITT RV24 ISO 2110
LAN	Ethernet Local Area Network	
	- Single CSMA/CD LAN (T/6211)	ENV 41101
	- Single multiple CSMA/CD LAN (T/6212)	ENV 41102
TP0	Transport Class 0	ISO 8073
TP4	Transport Class 4	ISO 8073
	Network layer addressing	ISO 8348/Addendum 2
CLNP	Connectionless Network Protocol	ISO 8602, 8073 DAD1

TCP	Transfer Control Protocol	De facto std
IP	Internet Protocol	De facto std
UDP	User Data Protocol of IP	De facto std
FDDI	Fiber Distributed Data Interface	ISO 9314
SNMP	Simple Network Management Protocol	Subset of CMIP DIS9596/2

1.2 Connectivity (layers 5-7)

Terminal Access:

VT100 VT220	Terminal compatible with DEC	ISO 6429 De facto std
TTY	Teletype compatibility	SIC-S1
Videotex	Videotex	CCITT R F300 CEPT T/CD 6.1 ENV 41501
Telnet	VT100/VT220 Terminal Access	De facto std

File Transfer

FTAM	File Transfer Access & Management - A/111 Single File Transfer - A/112 Positional File Transfer	 ENV 41204 ENV 41206
RA	Remote Actions	EWOS/ETG-003
MFTS	Multilateral File Transfer System based on NIFTP, Network - Independent File Transfer Protocol	SIC-S3, G3 Public UK Std
FTP	File Transfer Protocol	De facto std

Message Handling

TTX	Teletex	CCITT R F.200 T60, T72, T73, T90, T 91
MHS/X400	Message Handling System with Message Tranfer Agent (MTA) and User Agent (UA):	

	- Between a Private and Public Management Domain	ENV 41202
	- Between two Private Management Domains	ENV 41201
ILS	E-Mail product of INSEM project, based on MHS (envelope) and Teletex	
FAX	Facsimile telecommunications:	
	- Group 3	CCITT R T4, T10
	- Group 4	CCITT R T5, T6

Process-to-Process Communications:

NFS	Network Filing System	De facto std
RPC	Remote Procedure Call	ISO/DP 10148
XDR	External Data Representation	De facto std
RDA	Remote Data Base Access	ISO/DP 9579
CPIC	Common Program Interface for Communication	X/Open
X11-3	MIT (Massachusetts Institute of Technology): Windowing Protocol	De facto std
Directory	Directory	CCITT R X500
YP	Yellow Pages	De facto std

1.3 Portability

POSIX	Portable Operating System Interface	ISO 9945-1.2... IEEE TCOS 1003.x
MS-DOS	MicroSoft Disk Operating System	De facto std
UNIX	means complying with POSIX and X/Open	
UNIX	Operating System of AT&T	De facto std
COBOL		EN 21989
C		X/Open XPG3
CAE	Common Application Environment	X/Open XPG2/XPG3
XTI	X/Open Transport Interface	X/Open XPG3

1.4 Security

1.5 Storage Media

CD-ROM	Compact Disk Read Only Memory:	
	- Logical Level	EN 29660
	- Physical Level	EN 30149
Tape	Magnetic Tapes 1/2":	
	- Unrecorded	ISO 1864
	- Recorded at 6250 rpi, group-coded	ISO 5662
	- Labelling	ISO 1001
Fdisk	Floopy disk	ISO 7665, 7901, 6596, 7487, 8378, 8630, 8860

2. APPLICATIONS

2.1 Data

SQL	Structured Query Language	ISO 9075
ODA	Office Document Architecture:	
	- Q/111 ODA Basic Character Content	ENV 41509
	- Q/112 ODA Extended Mixed Mode	ENV 41510
SGML	Standardised Generalised Mark-up Language	ISO 8879
EDI	Electronic Data Interchange	ISO 9735
CGM	Computer Graphic Metafile	ISO 8632
DFR	Document Filing and Retrieval	ISO DP 10166
HP-PCL	Printer Command Language	De facto std

2.2 Graphical User Interface

X-Windows		De facto std
Motif	Graphical User Interface based on X-Windows	De facto std
MS-Windows	Graphical User Interface for MS-DOS of Microsoft	De facto std
PM	Presentation Manager, GUI for OS/2 of IBM	De facto std

2.3 Multilingualism

User Interface (terminals and printers)	
Character Set Requirements for the CEC	SIC-S11
CEC Requirements for a Multilingual Keyboard	ISO 8884 SIC-S6
Multi-user Computers	
8-bit code for Information Interchange, Structure and Rules Implementation	ISO 4873
Options:	
- G0 Latin Alphabet No 1, left part	ISO 8859-1
- G1 Latin/Greek Alphabet, right part	ISO 8859-7
- G3 Latin Alphabet No 1, right part	ISO 8859-1
- G3 Videotex	CEPT T/CD 6-1
8-bit single-byte coded character sets:	
- Part 1: Latin Alphabet No 1	ISO 8859-1
- Part 7: Latin/Greek Alphabet	ISO 8859-7
(Proprietary codification's are allowed if there is non-ambiguous coding or Greek and Latin characters (ISO 4873 Level 3))	
Personal Computers	
Native code--page CP 437 as well as proprietary codification's (e.g. EUROPA2, EUROPA-D2).	
Communications	
ISO 7-bit and 8-bit coded character sets, Code extension Technique	ISO 2022
Use ISO 2022 with the following options:	
- G0 Latin Alphabet, left part	ISO 6937-2
- G1 Greek registered set No 88	ISO 2375, Set 88
- G2 Latin Alphabet, right part	ISO 6937-2
- G3 Videotex	CEPT T/CD 6-1

CHAPTER 6

SYSTEM CONFIGURATION

6.1 SCOPE

The final step, of the three-step methodology prescribed in Chapter 1, in transforming user requirements into an operational system is to derive the actual hardware and software configurations from the system architecture discussed in Chapter 5. This transformation covers the selection of the hardware and software products in accordance with the architecture and the development of the functional area dependent applications software required to meet the user needs.

It will be appreciated that because of the vendor-independent approach adopted and the actual product selection being dependent on the user's particular requirements different configurations will in general emerge from the same system architecture. Consequently, as far as hardware configuration is concerned we shall only describe in this Chapter the types of hardware components in terms of their functionality, which together with the corresponding standards of Chapter 5 would lead to the appropriate hardware selection which can be made for the applications in question. The main emphasis in this chapter for the system configuration will therefore be on the application software configuration which will cover the software specifications derived from operational requirements, development and where possible the selection of appropriate COTS products.

6.2 HARDWARE CONFIGURATION

Specification and where possible selection of the hardware components of the nodes such as Terminals, Servers, displays, storage devices, communications devices as well as the specifications of the (LAN, MAN, WAN) which connect them together will constitute the hardware configuration. This will be considered in subsections below under the heading of "information processing" and "communications networks".

6.2.1 Information Processing

Information processing elements that are to be considered for C^3I system will mainly comprise terminals, PCs, workstations, servers, displays and storage devices.

6.2.1.1 Terminals

Terminals are the units whereby users interact with the C^3I system. Data entry, querying, monitoring, display of data and monitoring and control of the system functions are performed through terminals. They can be grouped into three categories with respect to their functionality:

- User Terminals: They are for end-users. Depending on the location of these terminals, they may be required to be tempest-proof. Security requirements, besides, impose constraints on the decision to have volatile memory for such terminals. The graphics capability, main memory requirements and input devices are determined with respect to functional areas where they are to be employed.
- Monitoring and Control Terminals: These terminals are for use by technical personnel responsible from operations and management of the C^3I system. These may contain specialized hardware and software capabilities depending on the tasks to which they are assigned.
- Display Terminals: These terminals are for display of graphics-data. They have high resolution graphics display capabilities.

6.2.1.2 Servers

Servers for each functional area are determined with respect to their performance requirements. A server may be dedicated to a single service or it may take over the responsibility of more than one function required by the user depending on the quality of service and availability of required performance in terms of user requirements. Servers provide a platform which support applications requiring database management, message handling, briefing support, graphics data display, window management, secure processing and communications, decision support services.

6.2.1.3 Displays

Briefing/De-briefing requirements, may it in civil or military, demand sophisticated display systems in a C^3I system.

Display technology is an area of intelligent support systems. Intelligent displays control what information is displayed, when it is displayed and how it is displayed. This may include whether voice, text or graphics are used most adequately for a certain situation. It may also involve user preferences leading to Human Computer Interaction (HCI) or Mixed Initiative (MI) in which the roles and the behaviors of the systems follow sets of protocols.

6.2.1.4 Storage Devices

In an information system fast retrieval and storage of information is key to the performance of the system. Storage devices to be used (configuration and

specifications) such as hard disks, floppy disks, CD ROMs, tape units should be selected taking into account access times, backup and recovery facilities, mirroring capabilities and security requirements needed.

6.2.1.5 Gateways, Routers and Bridges

Gateways, routers and bridges are internetworking devices. Today's corporations comprise specialized workgroups and a variety of networks may be in place to meet their differing needs. The interoperability of these workgroups is often limited by proprietary computer communication architectures that favor different types of LANs. The research and development division of a large company, for example may have chosen an Ethernet LAN to support DECNET applications, the business group, IBM's token ring and the technical documentation group, Apple Talk as the means for linking Macintosh microcomputers used for publishing applications.

Internetworking begins when the groups eliminate duplication of effort by drawing upon the resources of the others, improving efficiency and productivity. The connection of LANs spans to widely dispersed groups and, finally, the overall enterprise. The devices that feed traffic on these networks fall in the categories of repeaters, bridges, routers and gateways. These devices all interconnect networks and to various degrees their functions overlap.

6.2.2 Communications Networks

This section will describe the configuration of hardware related to information exchange services. In determining the communications configuration, the principal criteria are:

- System elements, interfaces, related standard and protocols and message and data transfer services provided will be homogeneous for any part of the system, i.e., users at any location of the system will be given identical capabilities of the system.
- Balanced survivability of the communications networks will be provided.
- The configuration at different command levels will be the same with the exception that the number and the scale of the system elements may be different.
- Communications will be secure at the level desired.

Network Traffic Analysis performed with respect to user requirements (See Annex 6-A) will generally dictate the use of FDDI (Fiber Distributed Data Interface) or ATM (Asynchronous Transfer Mode) technologies. The availability of standards and commercial products have an important impact in choosing the suitable technology.

Bandwidth requirements, services supported, compatibility with the future communications technology and trends and simplicity of architecture dictates the use of ATM technology but the main disadvantage of this choice is the lack of

related standards and software and hardware products at the time this book is written. However, the rapidly evolving technology and current trends show that ATM technology will be in widespread use very soon.

In the succeeding subsections, alternatives to LANs, WANs and MANs together with the pros and cons of each will be discussed.

6.2.2.1 LAN

There will be Local Area Networks (LAN) in each node system. The functionality required by the C^3I system is best met by client-server architectures. Bus and Ring topology networks are very suitable with clients and servers organized on the backbone. The importance of availability and performance requirements, however, makes the FDDI dual ring LANs more attractive. The configuration of a node of the C^3I system with emphasis on LAN is depicted in Figure 6.1. As shown in the figure, any user at a terminal may access an application running on any one of the servers transparently as long as the security checks hold.

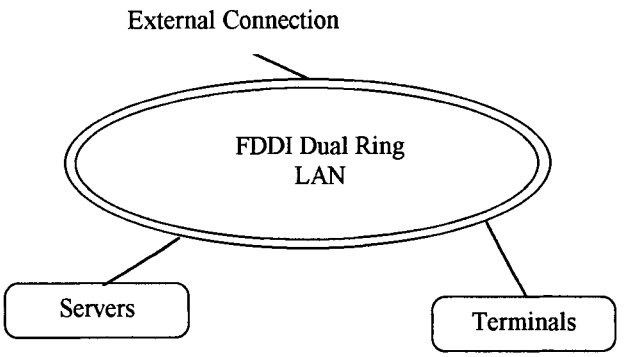


Figure 6.1 LAN of a C^3I Node

6.2.2.2 MAN

As the services like imaging, video conferencing, multimedia, high resolution graphics find application in the civilian/commercial environment, the need arises for interconnectivity having a traffic capacity to handle these facilities over an area to cover the boundaries of metropolitan area of a city or an entire university campus and hence a new network called Metropolitan Area Network (MAN). MANs are described in Chapter 5 where high speed networking technologies of FDDI and DQDB (Distributed Queue Dual Bus) are also compared.

6.2.2.3 WAN

The structure of the Wide Area Network (WAN) may have several alternatives in connecting the dual ring LANs of C^3I system nodes. In this section some configurations will first be presented and then the advantages and disadvantages of

these alternatives will be discussed. A possible configuration is to make one of the nodes such as a war headquarters a primary node and subscribe the LANs of the remaining nodes to the primary node LAN. The choice of the primary node in such a configuration depends on the organizational requirements such as the place where command and control activities are initiated in tension and crisis. In this configuration, LANs of the nodes are connected to the backbone LAN of the primary node via routers. To achieve network survivability, each ordinary node will be connected through at least two alternative routes to the primary node. In case connections will follow a ISDN integrated communications environment [6.6], these routers will support both FDDI and ISDN PRA interfaces. The connections between routers of the nodes will be through Dual Access Concentrators (DAC). This configuration is presented in Figure 6.2.

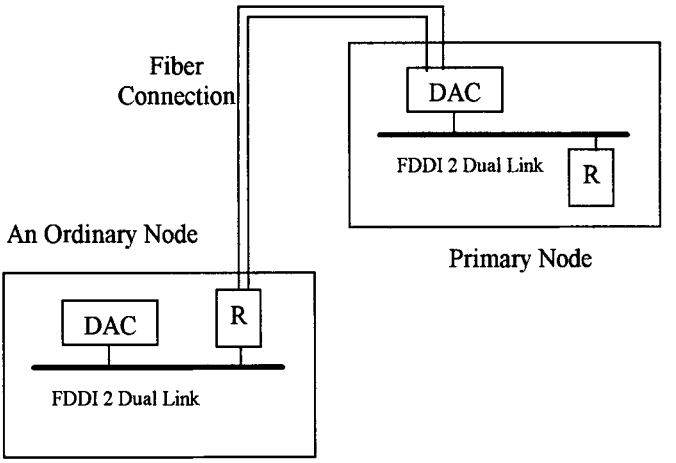


Figure 6.2 A Configuration with a Primary Node.

Another alternative configuration is to connect all the nodes through an FDDI backbone LAN. This configuration is shown in Figure 6.3.

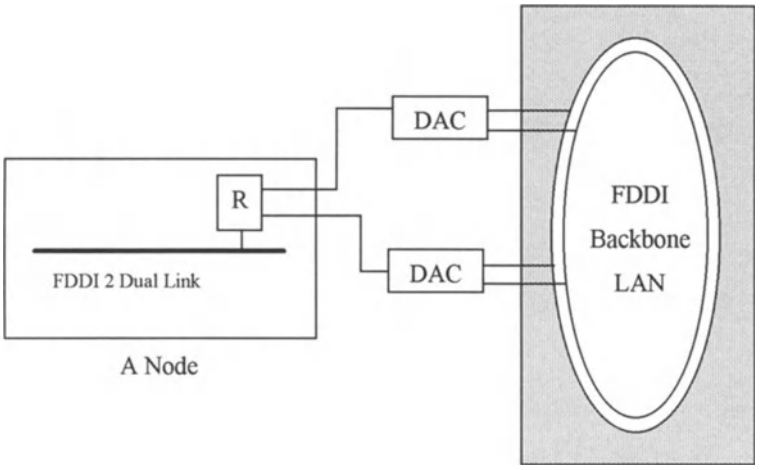


Figure 6.3 A Configuration with a Backbone.

As shown in Figure 6.3, a node is connected to the backbone LAN through two DACs for survivability reasons. The topology of the backbone is dual ring as is the topology of the nodal LANs. An alternative to this configuration would be to connect the routers of the nodes directly in a star-like topology as shown in Figure 6.4.

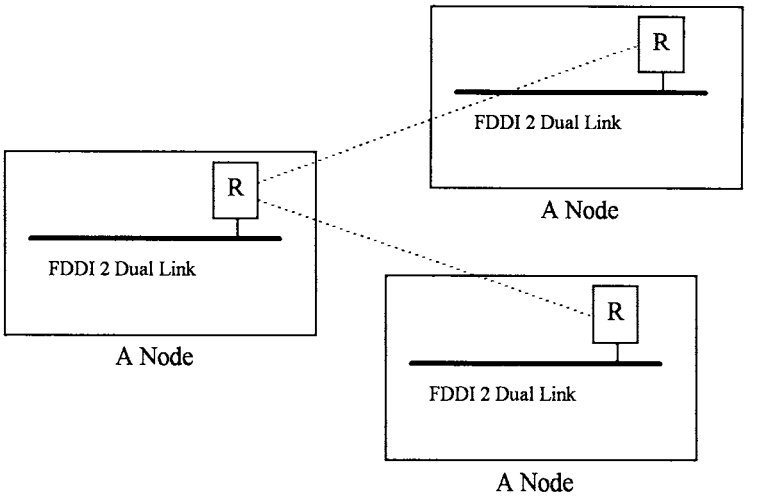


Figure 6.4 A Configuration with Direct Router Connections.

As shown in Figure 6.4, each router is connected to at least two other routers through FDDI or ISDN PRA interfaces. Routers will have the capability to perform dynamic routing.

The last alternative configuration shown in Figure 6.5 is to have ATM switches instead of FDDI LANs and to interconnect these ATM switches through a mesh topology WAN.

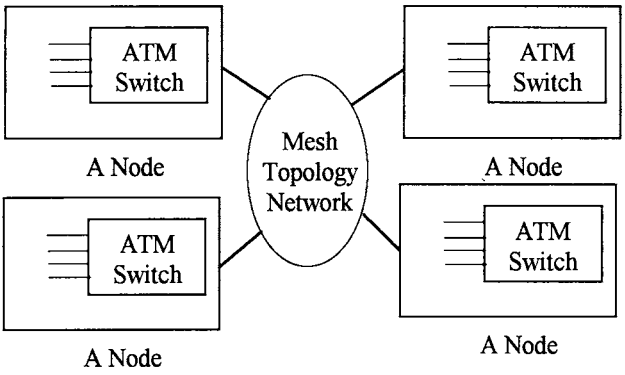


Figure 6.5 An ATM Oriented Configuration.

In all of these alternatives, a node lower in command hierarchy is connected to both its parent and to another node which is a direct ancestor.

Available bandwidth, services supported, efficient utilization of the available capacity, compatibility with future trends (such as Broadband ISDN), simplicity of architecture, limited number of system components makes the last alternative shown in Figure 6.5 the best candidate for configuration. The greatest disadvantage of this alternative, on the other hand, is the lack of related standards. The current trend, though, invites us to expect that they will emerge soon.

The alternatives shown in Figures 6.2 and 6.3 are in principle the same. The only difference is that the LAN of the primary node in the former provides for the functionalities of a WAN as well, whereas in the latter configuration a separate LAN is employed. If the main flow of data is in between primary node and the other nodes then the WAN functionality taken over by the primary node LAN will not cause an additional excessive traffic load due to communication requirements among Force Commands themselves. An additional advantage of the former configuration over the latter is the reduced latency over the communications network. A major disadvantage of these alternatives, on the other hand, is that they both are essentially based on the use FDDI LAN as WAN which, in turn, cause them not to be able to fulfill completely the elasticity and survivability characteristics.

The configuration presented in Figure 6.4 provides for the functionality required through routers and every router in a node is connected to at least two routers with dynamic routing properties. These routers, basically, imitate the WAN services provided by ATM switches. Availability of multiport routers supporting a versatility of interfaces and higher survivability offered makes this last alternative superior as a choice.

In general, every system component will be connected to FDDI Dual Ring LANs through DACs (Dual Attachment Concentrators). Each terminal output of DACs will be connected to one of the rings. This will ensure that the LAN management will be rationalized, i.e., depending on the network traffic load imposed by the system components, the number of DACs a server or a terminal will be connected and the number of ports such a DAC will have, will be determined.

6.2.3 Communication Services

C^3I system will be an open and distributed system extending over all the nodal systems at any level of the command hierarchy. Each command level is completely defined and is responsible for accomplishment of related command and control activities. As a result, there is a functional requirement to share and exchange information between commands. This requirement, in turn, brings into play the issue of interoperability among nodes of the C^3I system.

For interoperability, adoption of protocols in conformance with the ISO-OSI model and the corresponding ISO and other international standards and a common format for the data exchanged are the minimum requirements. The C^3I system nodes shall support at least the following services in an interoperable manner as explained in detail in Section 5.4:

- **Message Handling/Processing:** This service includes functions such as reception, submission, logging and archiving of all formal incoming and outgoing messages, retention of audit trails of all message transactions, translation between different message formats and origination and delivery of informal messages. WAN interfaces for message handling services will be implemented on a WAN gateway component.
- **File Transfer and Access:** File transfer and access services will be based on the ISO File Transfer Access and Management (FTAM) standard. FTAM is vendor-independent, i.e., it does not impose any constraints on the local file definitions or implementations. To achieve this, FTAM uses the concept of a virtual file store and allows file handling operations to be specified in terms of operations performed upon the virtual file store which are then mapped in a local way onto the local operations on the real filestores. FTAM provides functionalities such as transfer of all types of files to and from a remote node, manipulation of remotely held data and remote control of a filestore. It should be noted that for applications which only require to transfer small simple files X.400 is sufficient.
- **Remote Data Access:** Applications may require to access structured data that is stored in another node. The remote access and management of data held in structured databases is best accomplished through a different type of service than those previously described, namely, a remote database access service. File transfer and message handling would provide a less efficient and reliable mechanism to achieve this. For implementing direct access, manipulation and

replication of structured data across different nodes, the following aspects of heterogeneity have to be considered: Different data models and different database management systems. The first problem can be solved by standardizing the format and the semantics of the data subject to potential exchange. The second aspect is addressed by following a standard protocol for remote database access. ISO RDA (Remote Database Access) is an international standard defining a protocol allowing a client program to access a remotely located data manager. The so-called RDA SQL specialization deals with access to SQL DBMSs. For database replication, distributed DBMSs supporting two-phase commit (2PC) shall be used.

- **Electronic Mail:** Electronic shall be provided through COTS products A COTS package compatible with X.400 is preferred to facilitate E-mail and Message Handling integration. Integration between E-mail, word processing and business graphics packages will also be required.
- **VDU Conversation:** This service is based on ISO Virtual Terminal. It provides service for real-time conversation through user terminals supporting isochronous voice and image transfer.
- **Directory Services:** OSI X.500 directory services enable objects and their attributes to be defined locally. This allows users and processes to be defined and become addressable.
- **Network Management:** This service is for a robust management of the network connected devices. It will be based on OSI Common Management Information Protocol (CMIP).

6.3 SOFTWARE CONFIGURATION

The software requirements of the C^3I system can be classified into two categories with respect to their acquisition:

i. Software that can be acquired from the market as COTS and GOTS products. Examples of such software are Database Management Systems, Message Handling Systems, Graphical User Interface Builders, Compilers and so on. These software should generally be verified to meet certain criteria focusing on performance requirements and system characteristics including security aspects. This validation and verification can be performed within a test-bed environment.

ii. Software that need to be developed. They come under three categories depending on their complexity and application areas.

- **Database Applications.** Applications developed for storing and updating of data and presentation of data extracted from databases in formats desired by the user are in this category. Development of database applications are accompanied by the use of Computer Aided Software Engineering (CASE) tools to aid structured development and easy maintenance thereafter.
- **Graphics-Oriented Applications.** These applications generally include functions such as presentation of graphical data to the user and management of man machine dialog required by Man Machine Interface (MMI). In developing such applications, use of Graphical User Interface Builders (GUIB) and following object-oriented approaches have been shown to increase productivity and effectiveness.
- **Computation-Oriented Applications.** They are the applications developed for situation assessment, planning and decision making. The applications in this category are more complicated and need to be developed through prototyping.

6.3.1 Available Products Compatible with the Proposed Standards

Realization of products compatible with the proposed standards in Chapter 5 is a major step towards determining the system configuration. The order of preference over available products conforming to the standards can be given as COTS, GOTS and then NOTS.

It should be noted that products supplying operating system, man-machine interface, software engineering, data management and interchange, graphics, network and security services will constitute the core capabilities of the C^3I system (See Chapter 9).

When products which conform to the international standards are not available, they need to be conferment with the de-facto standards or they will be developed in-

house. It should always be kept in mind that the open system property of the system is completely dependent on the use of standard products.

6.3.2 Order of Software Development

Operational priorities and technical priorities collectively determine the order of software development.

6.3.2.1 Operational Priorities

This category of priorities is determined through a rank-order of the importance of software to be developed by the users of the operational system within functional areas. In other words, it is characterized by urgency of demand for applications. The primary condition for a C^3I system to achieve its objectives and to function properly is the reliable and fast flow of information in desired data rates from the sensors in predefined formats. As a result, the COTS software that are needed for the core capabilities such as Message Handling, Database Management, Briefing Support are ranked highest operationally. Applications related to decision support for functional areas such as intelligence, operations and logistics which are complicated and estimated to take more effort can be ranked next or they can be planned to be developed in parallel depending on the availability of experienced man power.

6.3.2.2 Technical Priorities

Technical Priorities are determined on the basis of dependency among software modules with respect to data flow requirements. For example a module waiting for the results produced by another module can be implemented later in time.

There may be cases where operational priorities and technical priorities are contradictory, in that, while one demands urgent implementation, the other may rank it lower in technical terms. In such cases, simulation techniques for producing the required input which has not yet really been produced by a module may be used.

An ordering of applications to be developed can be performed based on the data flow diagrams. The level of the diagrams used depend on the granularity of the applications ranging from a whole application to a subroutine or a function to be ranked technically. After deciding on this granularity which actually depends on the scale and accuracy of the lines of code estimations conducted for modules to be developed and the level of detail of the implementation schedule for these modules, affinity graphs are obtained. These graphs are directed graphs. The nodes in an affinity graph are units of applications to be developed and the edges denote the relationships such as read, write, update or create between two modules in such a way that an edge directed from node X to Y means that module Y needs the results produced by Y. These affinity graphs are topologically sorted to derive the technical priorities. To clarify the process, an example data flow diagram and the affinity graph derived from it are given in Figures 6.6 and 6.7.

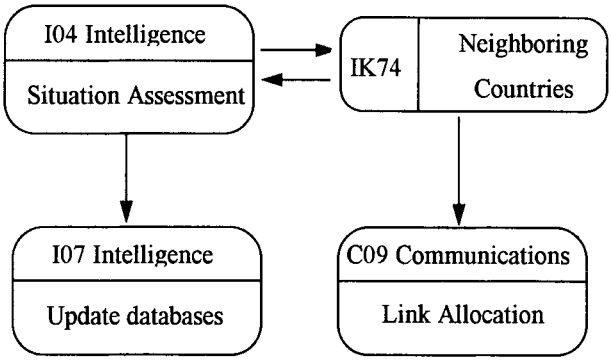


Figure 6.6 An Example Data Flow Diagram.

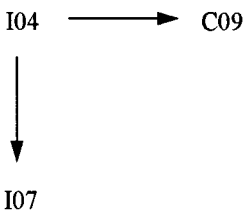


Figure 6.7 Affinity Graph Derived from the Example Data Flow Diagram.

Since data item IK74 is both read and written by decision/action I04 and it is only read by decision/action C09, C09 is topologically after the I04. A simple priority diagram for this example is given in Figure 6.8.

	Functional Area	
Priority	I	C
1	04	
2	07	09

Figure 6.8 Priority Diagram Derived from the Affinity Graph.

Figure 6.8 indicates the possibility that applications related to decision/action I04 are first developed and the applications involving decision/action I07 and C09 can subsequently be developed. If an affinity graph which cannot be topologically sorted as possible in case of decision/actions having feedback from each other or interacting with each other, these nodes can be merged or depending on operational priorities and ease of validation requirements an edge can be deleted biasing the opposite direction.

6.3.3 Classification of Application Programs

As stated previously application programs to be developed can be grouped in three categories as database applications, graphics-oriented applications and computation-oriented applications.

6.3.3.1 Database Applications

In developing database applications, structured development methods and CASE tools are used. Since a C^3I system is a distributed information system, exchange of structured information is key to the successful operation. One way of achieving seamless integration of local databases is to overcome difficulties stemming from semantic heterogeneity such as the difference in local data definitions at nodal databases which could be a troublesome task, if not impossible. A global Information Resource Dictionary (IRD) is required to be developed and maintained throughout the system life-cycle to have a common baseline for the structured data items. Such an approach facilitates the interoperability for component DBMSs as long as the underlying DBMSs support the same data model and the set operations with the same update semantics. Homogeneous distributed DBMSs are the most suitable candidates to be employed in C^3I systems. In case a specialized data management system is used for functional area specific activities, object registration tools, then, should be used and interfacing requirements should clearly be defined.

A framework for this database design is shown in Figure 6.9 [6.1]. The activity begins with a requirements analysis that defines the environment of the system and "elicits both the data and processing needs of all potential database user" [6.2]. The requirements study also specifies where the final system is expected to stand with respect to the objectives of a distributed DBMS. To reiterate, these objectives are defined with respect to performance, reliability and availability, economics, and expandability (flexibility).

The requirements document is input to two parallel activities: *view design and conceptual design*. The view design activity deals with defining the interfaces for end users. The conceptual design, on the other hand, is the process by which the enterprise is examined to determine entity types and relationships among these entities. One can possibly divide this process into two related activity groups [6.3]: entity analysis and functional analysis. Entity analysis is concerned with determining the entities, their attributes, and the relationships among them. Functional analysis, on the other hand, is concerned with determining the fundamental functions with which the modeled enterprise is involved. The results of these two steps need to be cross-referenced to get a better understanding of which functions deals with which entities.

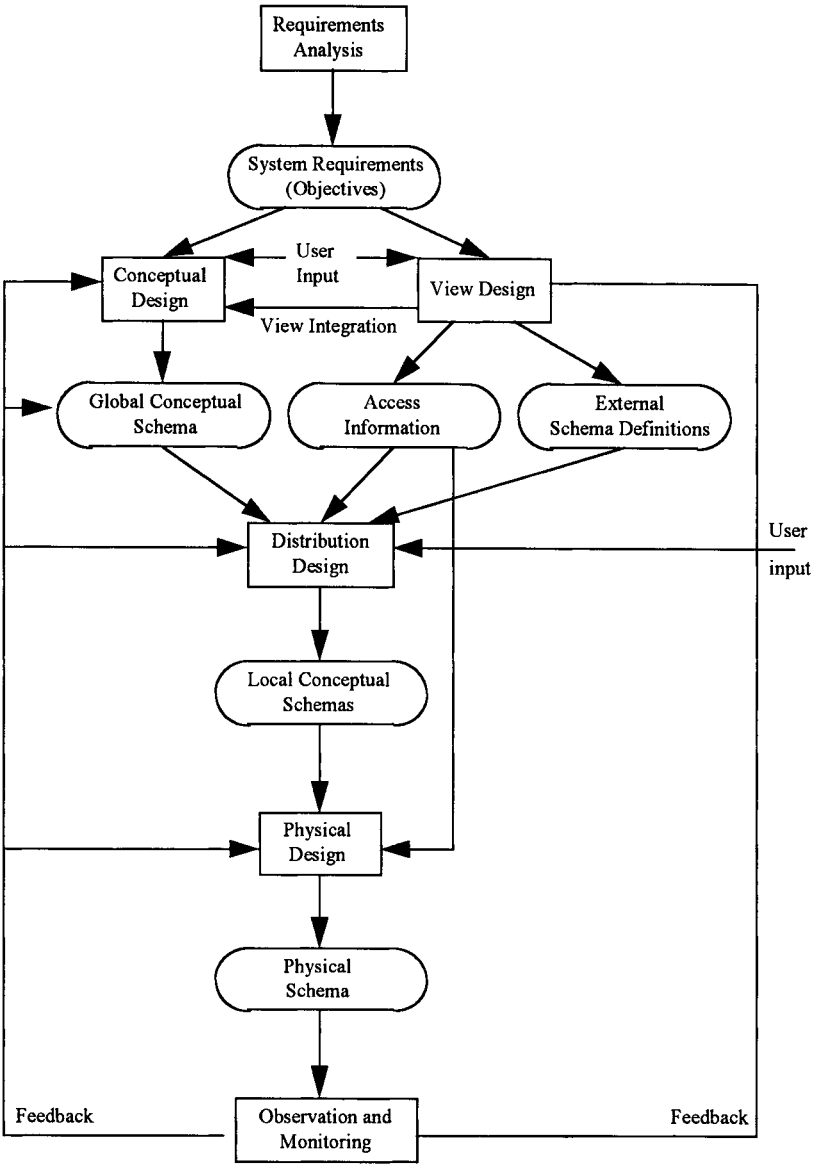


Figure 6.9 Top-Down Design Process

There is a relationship between the conceptual design and the view design. In one sense, the conceptual design can be interpreted as being an integration of user views. Even though this view integration activity is very important, the conceptual model should support not only the existing applications, but also future applications. View integration should be used to ensure that entity and relationship requirements for all the views are covered in the conceptual schema.

In conceptual design and view design activities the user needs to specify the data entities and must determine the applications that will run on the database as well as statistical information about these applications. Statistical information includes the specification of the frequency of user applications, the volume of various information, and the like. Note that from the conceptual design step comes the definition of global conceptual schema. We have not yet considered the implications of the distributed environment; in fact, up to this point, the process is identical to that in a centralized database design.

The global conceptual schema (GCS) and access pattern information collected as a result of view design are inputs to the distribution design step. The objective at this stage is to design the local conceptual schemes (LCSs) by distributing the entities over the sites of the distributed system. It is possible, of course, to treat each entity as a unit of distribution. Given that we use the relational model as the basis of discussion in this book, the entities correspond to relations.

Rather than distributing relations, it is quite common to divide them into subrelations, called fragments, which are then distributed. Thus the distribution design activity consists of two steps: fragmentation and allocation. These are the major issues that are treated in this chapter.

The last step in the design process is the physical design, which maps the local conceptual schemes to the physical storage devices available at the corresponding sites. The inputs to this process are the local conceptual schema and access pattern information about the fragments in these.

It is well known that the design and development activity of any kind is an ongoing process requiring constant monitoring and periodic adjustment and tuning. We have therefore included observation and monitoring as a major activity in this process. Note that one does not monitor only the behavior of the database implementation but also the suitability of user views. The result is some form feedback, which may result in backing up to one of the earlier steps in the design.

Another functionality required from a component DBMS dictated by the survivability characteristic of the C^3I system is replication. Survivability demands the availability of data in the presence of site and network failures. To protect against loss of data, other copies of data should be kept at other locations which may be completely different locations or other divisions at the same locations depending on the degree of protection to be afforded. A facility provided by distributed DBMSs, two phase commit (2PC) protocol, allows the definition and maintenance of replicated segments for data kept in databases in a consistent manner.

Given an interoperable and survivable environment for structured data, it becomes important to design and develop database applications in conformance with the user requirements. The very first step in doing so, is to be able to be definitive. The data acquisition methodology proposed in this book enables problems to be identified early on and quickly corrected. To help analysts, designers and implementors in developing database applications, CASE tools should be used. Although they are structured tools which ease the job, it is vital not to regard these tools as a recipe and not to be tempted to rely on a mechanistic technique and forget to think for yourself.

A successful C^3I system will have the characteristics such as goal oriented, rapidly developed, flexible and reliable. These characteristics define success when it has been achieved. There are many approaches to information system development depending on the needs of the organization. If the need owned is easily identifiable prototypes provide a rapid path to implementation.. Where the needs are more complex and inter-related a more formal process of expressing needs will be needed helping to identify global information needs and then to map onto these the requirements for gathering, managing, accessing and processing the information. In recent years, planning for complex inter-related systems has often started from an understanding of the fundamental processes in the organization So, underlying processes may be redesigned before even contemplating what information systems are needed to support them. But not all system developments begin with a green field. The legacy of existing systems make the recovery of design from existing systems necessary.

A meaningful approach is to use a hybrid approach which is a mixture of above approaches.

6.3.3.2 Graphics-Oriented Applications

Applications in this category will comprise mainly Man Machine Interface (MMI) services including specified interfaces, protocols and supporting data formats for implementing portable applications at the application/user interface and for communicating between the application/platform and the external environment. These services will provide client-server operations, object definition and management, window management, dialogue support and MMI security and management. Such services should be provided for both graphical and character-based display platforms. All graphics software used to provide two and three dimensional graphics development and viewing, interactive graphics and graphics data management should also provide at least a programming language binding conforming with the proposed standards.

MMI technology distributes computing power to the masses. Since operators are an integral part of the C^3I system, system engineers have to provide for enhanced computer user interaction through the development of cost effective graphics, color and animated displays. In a C^3I system, people are frequently presented with exhaustive amounts of data and must make critical decisions in brief time intervals. Because of this, MMIs are essential to the operational success of C^3I systems.

An MMI should be viewed as part of the system not as a software package between the system and the user. The traditional approach maintains both the user and the interface external to the system. This results in a fragmented operation in which an interface is frequently not well suited to the system or to the user and more often to neither. Another factor effecting MMI development is that software engineering principles are not given significant consideration in designing interfaces. Specifically, user specifications using the information hiding principle in an abstract interface need to be incorporated in the design of MMI software. An extension of MMIs is the concept of adaptive user interfaces. The idea of an adaptive interface is straightforward. Simply, it means that the interface should adapt to the user rather the user adapting to the system. The adaptive interface must encompass knowledge of the interaction, system, task domain and most importantly the user. Only with these types of knowledge will the interface be able to augment performance on an individual basis. The user's cognitive strengths and limitations must be incorporated into the system's knowledge base. An important know-how experienced in this field is that consideration of the cognitive characteristics of the user must occur early in the design cycle and not as an afterthought [6.4].

A decision maker's view of the system being used (the so-called mental model) affects joint decision-maker / computer system performance as much as more traditional factors such as screen design, type of interaction and input devices. The study of mental models has dealt specifically with what aspects of computers and humans affect mental model formation and has also shifted the primary goal human factors engineering away from task allocation toward task cooperation. Another approach of adaptive graphical user interfaces is the adaptation based upon the system's performance on the task. For example , an on-line telephone directory database can be reconfigured to make frequently called numbers easier to access. Regardless of whether or not AI truly places intelligence within computers, AI software systems treat human factors differently because of the designer's altered conception of computers. If mental models are great determinants of performance within systems without AI, they are likely to be huge determinants within systems containing AI, if only due to the perceived increase in computer intelligence. Moreover, systems designed from AI perspective affect joint human-computer performance in a second way. Computers within such systems have a conception of the user. This conception or image can be implicit as evidenced by knowledge-based systems that either passively assist the human decision-maker or assertively produce final decisions themselves, or explicit as with the intelligent tutoring aids and intelligent interfaces that infer the intend of the human by constructing a model of the human's cognitive processing. It is a tenet of the cognitive systems engineering approach to human factors that both mental models (human's of the computer and the computer's of the human) greatly affect performance and that a primary task of human factors engineer is to obtain matches between models and capabilities. Cognitive system engineering recognizes, therefore, that AI does not solve human factors problems; rather AI complicates them. The complication arises from system performance being affected by two underlying model and the great potential for model/capability mismatches.

The goal of cognitive systems engineering is to prevent sub optimal joint human computer performance by designing for mental model/capabilities matches. Thus, if

AI is introduced into a fusion system, it should be clear both to designer and end-user exactly which human capabilities are being assumed by the system whether they are human strengths or weaknesses and whether the possibility for human-computer cooperation is being eliminated, instigated or altered from some previous conception.

With the correlation area, the technology is moving toward supporting complete automation. This includes both increased computational power as well as new techniques such as the use of AI. This is appropriate in that the high data rates, which characterize modem sensor, far exceed the human's capability to keep up.

Complete removal of the person has a benefit. It allows the human to devote more resources to the fusion problems of determining the answers to the what and why. While a human is in the correlation loop, he is unconsciously being forced to browse through data. He has some appreciation of the overall situation or context because of his involvement in the correlation process. He may notice increased activity (e.g., through more correlation ambiguities in a given area. This can lead him to focus analytic effort in that area, such as looking for other data, which might indicate why the increase is occurring.

With the human out of the correlation loop, this cueing does not take place. Many of the current C2 systems are what might be called passive systems. Data are received, processed, and stored [6.4]. The human must specifically make request for information to find out if it exists. This is tolerable when the human is an active participant in the processing. When not, he has no cues to guide him as to where or when to request data.

The MMI actually becomes the system. The innards of the system becomes irrelevant to the operator. The MMI orchestrates the process, organize system contents and capabilities, and otherwise shield users from unfriendly interaction with complex data, knowledge and algorithmic structures. The use of expert systems which has Natural Language Processing as intelligent interfaces are necessary for problems that involve real-time or near-real-time decisions [6.4]. This feature brings a close interaction of the commander and computer inference engine. It is not possible for an expert system to operate as stand-alone consultation interference. The time-constraint of Natural Language Intelligent MMI (IMMI) brings two methods:

- The development of very fast Lisp processing environments, since Lisp is the most commonly used AI development language.
- The development of AI systems in a more "standard" and structured language, such as C or Ada. In languages such as these, the developer could write a compiler that translates the knowledge declared as production rules into a set of source code functions that are further compiled into object code at run time.

The second method causes the loss of flexibility that an interpretative Lisp environment brings, but it does provide a rapid execution of rules and easily expanded to include "non-symbolic" functions for computing primitive values.

Criteria for natural language systems that are usable and friendly to novices and experts alike:

- Syntactic coverage; that is, it should be able to parse dialogue syntactically.
- Task-oriented semantic coverage; that is, the interface should encompass a rich semantic knowledge of domain to compensate for its restrictions on legal inputs.
- Flexibility in presence of extra grammaticality; that is, the interface should be able to handle problems such as misspellings, transposed words, missing punctuation etc.
- Semantic resilience; that is, knowledge of the should be used to resolve ambiguities.
- User friendliness; that is, the interface should provide maximal assistance to the novice user and unobtrusive to the expert.
- Transportability; that is, the semantic domain knowledge should be separate from the interface can be used in different domains.

It is not sufficient for MMI to simply recommend actions, the MMI must be able to explain to a user reasons for its recommendations. With traditional expert systems this is done by displaying a trace of rules that resulted in the advice in question. With IMMI systems several factors combine to make this approach both inadequate and not feasible. First MMI advice justifications cannot presume the user is familiar with the heuristics embedded in the KB. Second, since good knowledge engineering is not based on the mimicking of human expertise, it is likely that the problem perspective embedded in the reasoning process will be somewhat different that the perspective of even domain-expert users. The third and most severe factor is the severe constraint imposed on the time available to MMI to justify its advice. For real-time applications MMI justifications must be understandable at a glance.

A method for acquiring user specific MMI is adaptive interface. Adaptive MMI means that the interface should adapt the user; rather than the user adapting to the system. There are two ways that a system can be adaptive. The first way is to leave the interface in a form that enables modification by the user if the behavior of the system is judged satisfactory once it is in operation. Although this may produce a better interface, it leaves the burden of adapting to the user. The second form of adaptation is dynamic adaptation by the system itself. An adaptive interface needs to have information that is generally not required or available to a static interface.

The concept of an adaptive interface does have critics:

- The user may not be able to develop a coherent model of the system if the system is frequently changing.

- The loss of control or feeling of loss of control that the user might experience.

- An increase in implementation time and cost.

On the other hand an adaptive interface can compensate for the inherent biases of the operator [6.4]. If carefully designed, the adaptive interface makes the system more useful to a larger number of people. Novices and experts can use the system more efficiently by providing them with the proper kind and amount of assistance for their individual needs.

An adaptive interface needs to include a knowledge-base that encompasses four domains:

- Knowledge of the user; that is, expertise with the system.

- Knowledge of the interaction; that is, modalities of interaction and dialogue management.

- Knowledge of the task/domain; that is, the ultimate purpose of the problem area and its goals.

- Knowledge of the system; that is the system's characteristics.

A technique of user modeling is to compare the use's knowledge with a domain experts knowledge. A user is to know something if the information or concept is used correctly. Furthermore the user is assumed to know additional concepts that must underlie those that are used.

If an adaptive interface is to provide help that is appropriate to the context as well as to particular user; it must be able to track the current human-computer dialogue. This requires some knowledge of how interactions are structured and what information may be implicit in them.

The interface works on several levels. Individual dialogue is analyzed. In addition, the whole session as a whole is tacked and dialogue is interpreted in the light of prior commands.

In most human computer interactions, a user tries to accomplish goals. These goals may be on several levels from the most immediate goals to the overall task goal. If a system is maximally supportive it must be able to assist the user in achieving these goals. In most cases, whether the dialogue is conducted in artificial language, users do not explicit state their goals. The system must be able to infer this information from the interaction. Only this way will an adaptive system be able to provide the most appropriate assistance.

The best computer support tool is most likely to be one that is easy to use and powerful in its capabilities. The ease of use of any adaptive system is closely tied to the input output capabilities of the system. As mentioned previously, natural

interfaces are inherently more adaptive than command interfaces because they do not require the user to adapt to the system's language.

The type of output should be dictated by the limits and capabilities of the user, the task type and the displayed information. The human operator has certain cognitive limits that must be addressed by the system output. Data should be displayed in a way that facilitates easy scanning., perception and interpretation. This may be accomplished in any number of ways. The data itself may be structured, the data may also be reconfigured on the screen to reflect its internal structure, color or highlighting can be used, or other more sophisticated pictorial graphics may be employed. The type of output is highly dependent on the nature and purpose of the information.

A number of guidelines, derived from human factors literature [6.5] are:

- **Mixed initiative:** An AMMI should reduce the cognitive burden on the user by sharing the initiative for the dialog. It should compile system and user state variables into a pattern which can be compared against templates for the goal-driven context, taking action which is appropriate for user support or for eliciting behavior from the user. The system should volunteer information to the user, as well as summarize and approximate responses in line with the perceived intent of an interaction. The system should learn from such encounters, permitting variability in response to similar situations that recur in the future. The user should be able explicitly to teach the system to modify such responses.

- **Vigilance Support:** To reduce the perceptual and cognitive load on the user, the AMMI should monitor input streams for conditions satisfying predefined situations to which the user should be alerted or which indicate a significant change in user context. Pattern matching should take account of variations which are within given probability bounds (a threshold for reporting). The user should be able to set up and modify situation definitions in accordance with his immediately perceived needs and problem-solving style.

- **Navigational support:** Transitions between one system state and another which are deterministic should be off-loaded to the AMMI. The need for the user to make choices among transitions which are not fully determined by the situation should be supported to the greatest possible extent by the AMMI, particularly for complex and variable task elements. The AMMI can prompt for navigation choices based on user history, problem-solving patterns, predefined templates, results of background processes, or context-keyed rule bases. The level of prompt should be modifiable by the user, providing more prompting for the casual user and less prompting as experience is gained. Prompting level should be context-specific, as well as user-specific, since a user is likely to become familiar with some tasks more quickly than others.

  The user should be able to change his navigation strategy in accordance with his experience and problem-solving style. Strategies which are advisable for

the novice user, such as extensive hierarchical menu and form-filling dialogs, should be bypassed by the experienced user.

- **Progressive disclosure:** The AMMI should present information structures as overall views, or notes of available information, with presentation of progressively greater detail under the control of user. The AMMI should have the capability of interpreting state variables in order to regulate the level of detail or resolution within a given context. The interpretive rules should be modifiable by the user.

- **Regulation of display surface:** The AMMI should use the contextual information as a cue to the user's attentional priorities and aid the user in structuring the display surface in an optimum manner. The user should also have direct control over the format of display surface, in accordance with his string of subgoals, problem solving style, and priorities. Particular configurations of the display surface should be able to identified by the user for retrieval under a symbolic name or by linking it with a contextual template.

- **Regulation of control surface:** Stored command sequences should be available for immediate execution by the user, or for system initiation on the basis of template matching. The user should be able to create such sequences and retrieve them under a symbolic name. The user should be able to modify selected command names, command formats, and objects operated on by commands.

  Relationships between single commands and command sequences, and the data which they test or transform, should be consistent with the user's model of domain. They should evolve the increasing user skill from simple associations of procedures and goals to complex strategies whose application is governed by selection rules. The AMMI should assume the burden of data type matching and should preclude inappropriate combinations of data and procedures. The user should be able to modify defaults at any time.

- **Metaphoric consistency:** Display and control surface organization, navigation options, context design, etc., should operate within a consistent metaphor that correlates with the user's conceptual models. That is, the system should provide a mapping between data processing concepts and the concepts within user's domain which permit functions to be accessed in a natural and intuitive fashion. This mapping should be consistent across tasks and data processing support functions. The metaphor should not exclude functionality, nor should it result in forced or unnatural mappings for any function. The metaphor should not require that efficiency be sacrificed to maintain it; shortcuts and enhancements for advanced users should be subsumed by the metaphor. The system should adapt to the user's evolving view of the metaphor, and anticipate corrections required due to its connotative characteristics.

- **Short-term memory support:** The machine's memory must be tapped to provide every possible support for the users limited and fallible short-term

memory. The user should be able to exploit the systems memory by freely storing scratch pad files, maintaining a personal database, modifying help materials, setting up remainders, and creating customized templates/alarms.

- **Maintenance of user context:** The user's context should be monitored and preserved, and he should be able to restore context after an interruption. The user should also be able to retrieve a given generic context under a symbolic name. The system should maintain tracings of the history of the user contexts for identifying repetitive tasks that can be more fully automated, and for analyzing task sequences for more efficient pathways.

- **Support for context customization:** The user should be provided with models for processes which can be copied and modified in a new context to generate additional strategies. In general, copying / altering existing entities in the system is preferred to forcing production from scratch.

- **Learning acceleration:** Training should be embedded. The AMMI should explicitly increase the user's levels of expertise while the user is performing his tasks, although such training should not interfere with task execution. The user's skills should be constantly reevaluated, and training or help materials keyed to the user's current level.

- **Error recognition:** The system should monitor errors in terms of probabilities and signal corrective action. Different contexts require different loading of error parameters. Remedial action should be prompted on the basis of an understanding of the source of the error and probable intent of the user, given the context.

6.3.3.3 Computation-Oriented Applications

The applications in this category lies at the heart of the C^3I system. Today Command and Control (C2) systems are complex organizations of people and equipment which have to function in a coordinated way to achieve their objectives. Man has already amplified his muscle power by using mechanical systems, his senses by using electromagnetic and acoustic devices, his ability to communicate over long distances by using radio, and his calculating capability by using computers but thinking is still the preserve of the human brain [6.5]. This is a limiting factor in several aspects: The number of brains that can be brought to bear on a problem is limited especially by cost; the level of experience is limited by the amount of training and the term of service; the communication ability of the brain is limited by fairly low bandwidth input and output; the brain requires a special environment in which to operate and the thinking power of brain while impressive is rather slow.

The aim of the C^3I systems is not to replace the human element, but to augment specific aspects of the brain to improve overall effectiveness. In order to achieve this a C^3I system must provide:

- An appreciation of the current situation,
- A communication system,
- Immediate control,
- Ability to distribute intelligence,
- Up-to-date logistic information.

The part of the C^3I system aiding in decision making can be structured as shown in Figure 6.10.

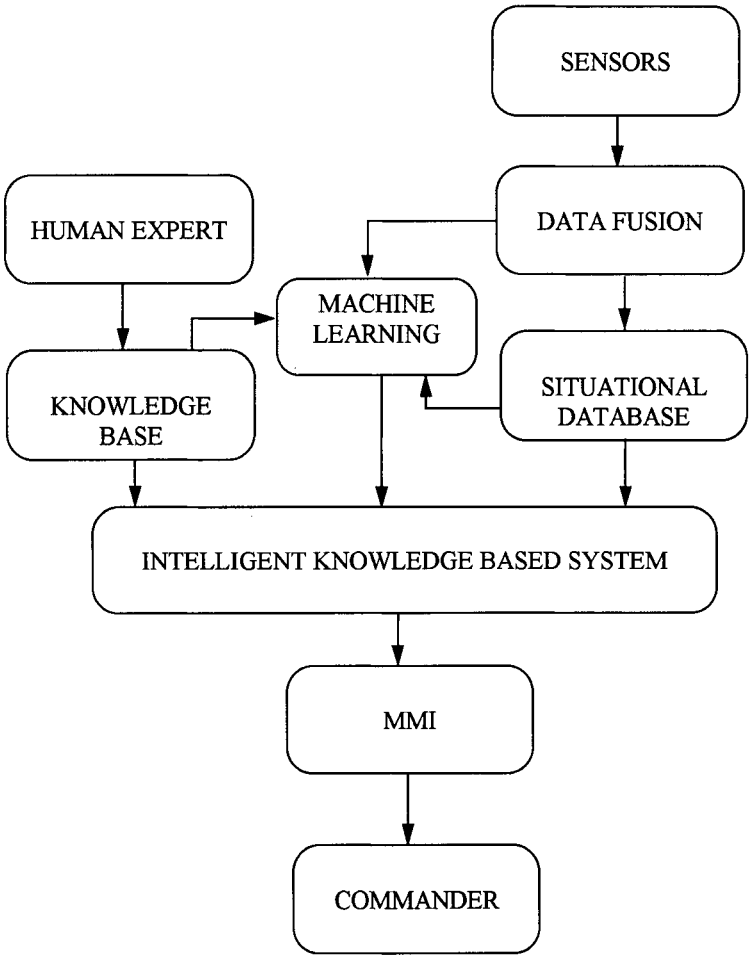


Figure 6.10 Decision Support in a C^3I system

The process depicted in Figure 6.10 can be decomposed into the general areas of:

- Sensor and data understanding: Processing and fusing raw data into descriptive information.
- Situation understanding: Inferring environmental descriptions from information observable.

- Planning and control: Designing actions to satisfy goals of controlling or effecting the environment.

C^3I systems require interaction of the following two primary functions [6.5 and 6.10]:

- **Data fusion:** Data fusion collects information from a variety of sensors and sources to develop the best possible perception of the situation. The fusion process includes collection, association, aggregation and merging of data to create and display current and past situations.
- **Decision support.** Decision support performs the creation and quantitative evaluation of alternative estimates of the real situation and responses available to the decision maker. The paradigm for C^3I decision-making contains four elements, namely, Stimulus, Hypothesis, Option and Response. Stimulus covers the initiation of the decision-making process which is the data fusion function providing information on the current situation with associated uncertainties. These uncertainties result from lack of sensor coverage, ambiguities in reports, conflicts in reports, or inadequacies in measured data. Because of uncertainties in correlated sensor data, the data fusion and decision support processes must jointly create hypotheses that are best candidate real-world military situations that explain the fused data. For each hypothesis, options are created and expected military results are modeled. These options and results are provided to the human decision maker for selection and action. Finally the response is the selected response which is taken by the forces under command. This creates new stimuli to overall system.

The cognitive system engineering unifies the distinct field of psychology, information science and system engineering. Presented below is a set of findings from the cognitive sciences, the findings are organized around the two exemplar generic domains of inference making and decision making:

Inference-Making Findings:

- Inference making performance is task dependent.
- Human Information Processors (HIP) have poor insight into their own inferential processes.
- Cognitive feedback can improve task performance significantly.
- HIPs are susceptible to nonlinear-based deception.
- HIPs use simple and complex casual schema to explain behavior and make inferences.
- HIPs often reason from experiences, analogies or cases.
- HIPs use plausible scenario generation to test and generate hypothesis.
- HIPs relay on cues-to-casualty to explain current and future behavior.
- HIPs use less information than they use to make inferences.

- Experts are susceptible to poor insight about their inferential processes as novices.
- HIPs are unsure about how to optimize data, information, and knowledge in inference making.
- HIPs are prone to cognitive reductionism to reduce the workload and overall mental effort.
- The perception information is not comprehensive, but selective.
- HIPs tend to underemphasize base-rate information.
- HIPs weigh the importance of information on the basis of its perceived casual meaningfulness not on its statistical diagnosticity.
- HIPs are susceptible to the availability bias or the tendency to recall recent or highly publicized events.
- HIPs selectively perceive data, information and knowledge on the basis of experience.
- HIPs are susceptible to confirmation biases that filter incoming information according to preconceived ideas and views.
- The way information is presented often determines how it is perceived.
- HIPs tend to be conservative inference makers.

Decision-Making Findings:

- Human Decision Makers (HDM) tend to simplify decision problems by setting outcome aspirations.
- HDMs often choose the first decision alternative that satisfies the outcome aspirations.
- HDMs often simplify decision problems by only considering small number of alternatives.
- HDMs use analogies to generate and compare results.
- HDMs weigh criteria to rank-order decision alternatives.
- HDMs selectively perceive data, information and knowledge; focus on confirming (versus disconforming) information and tend to anchor their judgments.
- HDMs tend to attribute decision outcomes to change or reduce the complexity of the problem (not their own decision making deficiencies).

Because of the above problems, the capture of analytical knowledge is difficult. Conventional software systems engineering methods have failed to provide designers with the flexibility needed to iterate on requirements data and definitions. The prototyping strategy has been proposed as a solution to the systems design and development bottleneck. Prototyping calls for the modeling of user, task, and organizational requirements in the form that users can evaluate. Prototypes are intended to reduce the gap between what users need and what the interactive system deliver. Storyboarding is another prototyping technique that borrows from requirements analysis and simulation. Storyboards are interactive screen displays of system functions. They are inexpensive shells of system performance, i.e., they are not usually driven by working software but rather screen displays of functions- that

when validated will subsequently be programmed. Storyboarding permit system sizing. Sizing involves the specification of the data and knowledge bases necessary to drive the system to be the specification of the analytical methods that might drive the system, the identification of key software systems engineering issues and the development of alternative interface strategies.

The Knowledge Base (KB) is a part of the system which contains a map of C2 expertise which operates on known conjectured or hypothesized cause and effect relationships, implications, heuristics, inferences and operational plans. Typically a KB system contains rules and procedures [6.5].

In C2 applications the conventional knowledge base system based upon the construction of rules of if / then clauses is inadequate since situation evaluation frequently involves uncovering situations that complex evolving sets constraints rather than simply evaluating a set of fixed truth conditions.

The rule format proposed for situation analysis based upon predicate calculus which enables the representation of knowledge in a descriptive manner. By using this general form, we achieve both rule standardizations for C2 systems involving thousands of rules and effective representation. This approach places all qualified variables in front of all predicates. This provides maximum processing efficiency. Since the problem is then well known generate and test format. The situation analysis rule is, therefore, in the shape of a number of conditions; if these are satisfied, an indicated action takes place. Such a rule based system requires interpretation of confidence limits used in the conditions such as near range number which gives greater flexibility in system assessment.

An alternative methodology is to allow for the existence of non-expert solutions narrowing and partitioning the solution space into a set of skeletal plans based upon apriori scripts that capture the heuristics used by expert planners.

Although the skeletal plans do not produce optimal solutions, they produce at least one skeletal plan for any given problem. A skeletal plan is initially matched to the domain state, and its current domain state effectiveness is measured; further refinements of skeletal plans then take place. At each iteration, evaluation, for effectiveness and satisfaction of constraints is carried out.

Two of the major abilities that a human operator brings to the task of controlling a system are the use of a flexible body of knowledge and the ability to integrate synergistically information of different modality obtained through his senses. The increasing use of knowledge-based expert systems is an attempt to capture some aspects of first ability; current research in multisensor integration is an attempt to capture, and possibly extend to additional modalities, aspects of this second ability.

Multisensor integration refers to the synergistic use of the information provided by multiple sensory devices to assist in the accomplishment of a task by a system. An additional distinction is made between multisensor integration and the more restricted notion of multisensor fusion. Multisensor fusion, refers to any stage in the

integration process where there is an actual combination into one representational format.

- Multisensor data fusion can be viewed as a hypothesis generating process for C^3I assessment and decisioning. The MSDF has at least for sub problems:

    - Alignment process and position data fusion; checks if sensor data refers to some entity and representation.

    - Association process or comparison problem; checks if two representations into one for same entity.

    - Aggregation into a more abstract representation; elementary form of situation assessment

The purpose of external sensors is to provide a system with useful information concerning some features of interest in the system's environment. The potential advantages in integrating / using information from multiple sensors are that the information can be obtained more accurately, concerning features that are impossible to perceive with individual sensor, in less time, and at a lesser cost. These advantages correspond, respectively, to the notions of redundancy, complementary, timeliness, and cost of the information provided by the system.

Redundant information is provided from a group of sensors (or single sensor over time) when reach sensor is perceiving, possibly with a different fidelity, the same features in the environment. The integration or fusion of redundant information can reduce overall uncertainty and thus increase the accuracy with which the features are perceived by the system. Multiple systems providing redundant information can also serve to increase reliability in the case of sensor error or failure.

Complementary information from multiple sensors allows features in the environment to be perceived that are impossible to perceive using just the information from each individual sensor operating separately. If the features to be perceived are considered dimensions in space of features, then complementary information is provided when each sensor is only able to provide information concerning a subset of features that form a subspace in the feature space, i.e., each sensor can be said to perceive features that are independent of the other sensors; conversely, the dependent features perceived by sensors providing redundant information would form a basis in the feature space.

More timely information, as compared to the speed at which it could be provided by a single sensor due to either the actual speed of operation of each sensor, or the processing parallelism that may be possible to achieve as part of integration process.

Less costly information, in the context of a system with multiple sensors, is information obtained at a lesser cost when compared to the equivalent information that could be obtained from a single sensor. Unless information provided the single sensor is being used for additional functions in the system, the total cost is single

sensor should be compared to the total cost of the integrated multisensor system [6.4].

General approaches to multisensor integration and fusion [6.4]:

- Paradigms and frameworks for integration
  - Hierarchical phase-template paradigm.
  - Neural networks.
  - Logical sensors.
  - Object-oriented programming.

- Control structures
  - The NBS sensory and control hierarchy.
  - Distributed blackboard.
  - Adaptive learning.

- Sensor selection strategies
  - Preselection.
  - Real-time selection.

- World Models
  - The Multisensor kernel system.
  - The NBS Sensory system.

- General fusion methods
  - Weighted average.
  - Kalman filter.
  - Bayesian estimation using consensus sensors.
  - Multi-bayesian
  - Statistical decision theory.
  - Shafer-Dempster evidential reasoning.
  - Fuzzy logic.
  - Production rules with confidence factors.

Intelligent Knowledge Based Control Systems (IKBCS) can be used as decision / planning aiding techniques that allow the user to concentrate on the options offered and to determine the most appropriate response.

The IKBCS will accept data from a variety of sensors, databases and intelligent sources will correlate, combine or fuse them, will lialise with the KB, via optimal search or inference rules, and will then perform plausible or rational decision tasks from which modification of the existing KB would occur consistent with the new data.

The intelligent knowledge base control system actions are to process input information from situational database, and to interact with the knowledgeable rules

to procedure the situational analysis and to provide with the user input / output information (including explanations), of all which must be processed in real-time.

Monitoring the situational database IKBCS assesses the current situation, recognizes plans and makes predictions. Using this knowledge IKBCS makes tactical planning, mission planning, resource allocation, route planning and asset management.

Planning can loosely be defined as the ordering of actions over time. To generate a plan, the system apply a KB of goal and actions. These KBS define the conditions under which goals and actions are feasible, and the constraints that needed to be checked to test their feasibility. A significant portion of planning involves the generation and testing of constraints. A successful plan is the one that satisfies the defined plan objective without violating any constraints. Reasoning about expendable resources, for instance, can be handled by posting constraints on the minimum allowable level on that resource and constantly checking the plan against this constraints [6.4].

Because planning is such a vital part of the C2 function, the hypothetical promise of intelligent computer planning systems and the actual slow pace of progress in the research and development of these systems is of obvious concern. Foremost among the contributions that could be made by reliable and timely AI based planning systems would be the ability to monitor complex situations where large quantities of data need to be assimilated quickly and to project outcomes of possible courses of action in dynamic environments. AI to data has been to deliver very little in the way of successful contributions in this critical area. Systems have been limited to the simplest rule-based paradigms, and so their ability to cope with dynamic, uncertain, and adversarial situations is severely constrained.

Relatively few C^3I systems are built to support senior command-level decision process. This is, in part, because effective decision making at senior levels usually involves a decision process that calls upon the decision maker's broad experience and knowledge base. AI systems on the other hand, have usually been limited to areas where the knowledge required for the problem solving is both well defined and limited in scope.

To the extent AI based techniques can be in the position to provide significant support to senior decision makers, it must serve primarily in the role of a local advisor or critic. That is, while an AI system may not be able to call upon a senior commanders extensive experience and knowledge base to generate decision options, it may be in a position to evaluate important options from the perspective of limited sub problems.

Many prototype C^3I systems treat AI as developing a problem solving model. In contrast, operational experts systems are driven by a difficult domain problem and view AI as a contributing technology. Whether AI develops a model of human problem solving is secondary to its ability to solve the problem. Operational expert systems contain far less AI percentage-wise, than laboratory systems.

To build an aid of this type, two design problems must be addressed. The first is that of communication with the decision maker. Since it is still the decision maker's role to control the option generation, the decision generation must have embedded within a simple and natural mechanism for transcribing and representing user generated options [6.4]. This may quite difficult for decision problems such as planning maneuver campaigns, where an option may be a complete concept of operations or a plan that involves many factors, sequencing considerations, levels of detail, etc.

To achieve real-time operations using conventional on Neumann computer architecture's, task scheduling based upon priorities is necessary. Satisfaction of all goals or plans in real-time is generally not feasible; then the choice of what tasks have greatest urgency and priority is significant. Partial goal fulfillment and the associated representation and reasoning about the effects of goal architecture has attracted little attention.

As C2 systems require a large Knowledge Base, there are two primary constraints:

- Information storage limits
- Execution time

Distributed IKBS as well as parallel processors offer the option of several independent systems operating on various aspects of overall problem communicating via network.

The centralized control system (master monitor) provides concurrent overall subsystem KB system management through a communication network. It accesses other KB's situational databases, users, and strategy monitors. The strategy or objective control system provides, in order of priority, the tasks or rules determined by the current situation that require processing in the situation assessment system.

The solution procedures are based upon human expert guidelines and general analytical or logical processes (i.e. Fuzzy Logic); these procedures provide links between the tasks and suggest what should be done next. Solution time is dictated by complexity rather than the size or dimension of the rule-base.

An adaptive priority urgency should be adopted that updates a priori goals / plans on the basis of current situational database. An important architectural attribute would be an interrupt system that enabled the system to adopt new goals without misinterpretation of decision goals.

Situation assessment or rule processing is essentially stochastic and can be executed through a probabilistic data structure priority queue. The rules are probabistically selected from the queue, based on the relative priority of the subject contained within the rule. These priorities act as dynamically varying operating constraints upon the control system, since they can be altered by the control system itself via control rules in the strategy control system.

New sensor or intelligence data is filtered and added to database the significance of the data is determined by data fusion and correlation with other data, and then by application of interpretation rules in KB.

The significance of the new data is achieved through a progressive forward and backward chaining process to identify the most significant or the highest priority situation data.

Following data acquisition and situation interpretation, suggested actions, strategies or conclusions are given to the user via an MMI, together with explanations.
The decision justification is a consequence of complex logical inference chains in the KB under the control of the strategy control system.

The attribute of explaining the reason behind advice / status / decision recommendations given to user by the KB system is essentially to the human decision maker, since he can directly assess the justification for recommendations before implementing the actions.

Parameters that should be included in C^3I Knowledge Base Systems are [6.5]:

- **Search and query:** The C^3I system must support the search / query function in order to reach the information needed. The search / query function must work in the dynamically changing information structure of the C^3I system. There are two broad classes of search / query: content search, and structure search. In content search all information is considered as independent and are examined individually for a match to the given query. In structural search the internal structure of the C^3I system is searched. Both of these functions are necessary for a C^3I system to enable a commander to join the correlation loop, with browsing through the data.

- **Composites:** The C^3I system must support the composites, for clustering the information and hiding unnecessary details. These composite information clusters could represent the new formation of the current information or different aspects of the information already used.

- **Virtual structures:** Virtual structures are for dealing with changing information. A user in the very early stages of working with a particular set of information may not sufficiently understand the content and structure of that information. Knowledge about the critical dimensions of the idea space, the characteristics which distinguish one idea from another, and appropriate naming schemes develops over time as the user becomes familiar with the information. As the user's knowledge of information space evolves, previous organizational commitments became absolute.

- **Computation in C^3I network:** Computation built into C^3I system is likely to be more efficient, especially when that computation involves extensive access to information in the network. In contrast, an external computational engine is

less restricted since no commitment to a particular computational engine needs to be made when the system is to be implemented.

- **Versioning:** A good Versioning mechanism will allow users to maintain and manipulate a history of changes. It will also allow users to simultaneously explore several alternative configurations for a single network.

- **Support for Collaborative Work:** Creating annotations, maintaining multiple organizations of a single set of materials, and transferring messages between asynchronous users are the kinds of activities that form the basis of any collaborative effort. Although simultaneous multi-user access to the information is provided this is not adequate. For example; A mechanism for notifying users of important actions to other users is needed. The C^3I needs to drastically improve support for to disparate but interrelates areas: The mechanics of simultaneous multi-user access to a common network, and the social interactions involved in collaboratively using a shared network.

  In the area of support for the social interactions involved in collaborative use of a shared network, the critical notion is *mutual intelligibility* [6.5]. In a collaborate effort, each participant must have some degree of understanding of the actions and intentions of any collaborators.

- **Extensibility and Tailorability:** In order to add new functionalities or tailor existing functionality to better match the exact requirements of the applications extensibility and tailorability is needed.

  The C^3I system is characterized by [6.5]:

  - **Complexity:** Complexity refers to the number of goals of the system, the number of processes that must be controlled to reach the system's goals, and the number and descriptions of the means available by which the processes can be controlled.
  - **Delays:** Delays, such as information delays or delays in the command system, any or may not affect the processes and the system goals.
  - **Characteristics of the processes and means:** Characteristics of the processes and the means can affect the controllability of the processes. For example, when a process with exponential growth is to be controlled with a linear process.
  - **Rate of change:** Rate of change of both the processes to be controlled and the means may affect controllability of the system by required nature of feedback information of the situation.
  - **Delegation:** Delegation of decision-making power by imposing a hierarchical organization for control.
  - **Feedback quality:** Feedback quality can affect the ability of the commander(s) to perceive the situation clearly. If the feedback information is of low quality, in the sense of being imprecise or uncertain, the inferences concerning situation is affected.

- **Probabilistic or deterministic aspects:** The extend to which system responses are predictable in advance.

The aim of Artificial Intelligence (AI) which is very important as a basis of decision making is to understand and to model human comprehension mechanisms. There are many disciplines, called cognitive sciences such as psychology, linguistics, and neuroscience which are working in this direction. But the main difference is that AI expresses its models and theories about knowledge and reasoning by programming computers. AI uses different approaches such as Expert Systems, Fuzzy Logic, Neural Networks, Logical Programming, Multi-agent Planning Methodologies.
Conventional expert systems which usually follow a slow question and answering procedure cannot meet real-time requirements. Some characteristics of a real-time expert system may be listed as :

- Input data of large amounts such as track data should be accepted automatically.
- The automated interpretation of data is required to answer highly qualitative questions.
- The response time is predefined by the environment.
- Facts change during processing.
- Sequence of processing changes with respect to the urgency of alarm levels.
- It should have an interrupt capability.
- It should be able to run continuously.

In order to meet these requirements the architecture of conventional expert systems have to be modified. The variations are:

- controlling the depth of the inference process,
- adapting the structure of the knowledge base to the real-time requirements,
- reducing the amount of data by pre-computing,
- increasing the performance, e.g., by parallel processing.

The criteria for the assessment of C2 tasks as a promising application of AI can be given as follows:

- If there is a conventional solution then it should be done conventionally. However, certain additional factors such as flexibility or growth potential may suggest the use of a AI based approach.
- Areas such as diagnosis, planning, scheduling, monitoring and decision support have proven to be suitable for AI approach. If a problem fits in one of these categories, the AI approach looks promising.

As to the problem domain, the following should be explored.

- Does the problem require expert knowledge, experience and capability?
- Can the problem be represented symbolically?
- Can sub-optimal solutions be accepted?
- Is the explanation of the system's solution process highly desirable in order to obtain confidence and clearness?
- Is knowledge available (experts, manuals, written regulations/procedures, etc.)?
- Do suitable knowledge representation mechanisms and inferencing strategies exist?
- Can input data and parameters be provided?
- Is it possible to provide a suitable simulator or test environment?
- Would knowledge and experience be lost if not kept alive in the system?
- Is the same knowledge and experience needed at various places?
- Is the AI application area involved still in the stage of research?
- Is time available for development very limited?
- Is the problem part of a new application domain or one which lacks fundamental understanding?

Two keys for an effective development process are use of evolutionary prototyping and early involvement of end users.

The development of cooperative systems that combine conventional and knowledge processing modules and techniques such as blackboarding for the integration of modules with different inferencing mechanisms allow independent subsystem development and evolution.

A major impact of the AI on C2 systems depends mostly on integration issues which are addressed in efforts to develop knowledge base sharing, knowledge representation languages and knowledge base / data base systems.

AI application areas of expected high payoff still carry risk due to lack of mature technology and experience. The important areas of risk are real time requirements, interactive operations, adaptive and dynamic reaction.

C^3I application problems investigated are usually rather hard to solve, knowledge intensive and not well structured. The lack of knowledge in the domain investigated and the lack of appropriate knowledge acquisition methodologies must not be blamed on the AI discipline. Knowledge acquisition methodology proposed in this book improves the process of knowledge acquisition.

In the light of above discussions, the recommendations for the design and development of C^3I applications based on AI techniques are presented below:

- Choose AI applications where successful R&D and/or mature solutions already exist.
- Do not apply AI where conventional techniques do the job.
- Establish an incremental software development cycle including adaptive rapid prototyping if necessary.
- Organize and clearly define user involvement and involvement of domain experts for problem definition, knowledge acquisition and validation and verification.
- Develop measures of merit of AI together with the user and apply them at well defined validation and verification milestones.
- Any C^3I system applying AI will become a hybrid system of conventional and AI oriented software. The interface problems, therefore, should be identified, defined and solved a s early as possible.
- Make sure the scientist/software engineer really concentrates on the solution of the problem and not on peripheral issues such as graphics, MMI, tools, DBMS, simulations etc.
- Do not let novices do the work neither in technical implementation nor in the management.
- Expect higher investment/effort at the beginning; the payoff comes with the operational use. Think in life cycle cost categories.
- Initiate standardization activities in the responsible organizations. Derive standards (e.g., class libraries, tools) from existing and evolving commonalties.
- Take into account the use of some new techniques such as Procedural Reasoning Systems (PRS), Neural Networks (NN) and Case Based Reasoning (CBR) Systems. Both procedural systems and case-based systems along with neural networks are capable of dealing with uncertain and incomplete information. This differs from rule based expert systems and classical generative planning systems that require complete and certain knowledge of the world. The use of PRSs in dynamic real time systems includes replanning, plan repair and interactive scheduling when complete reasoning is not possible. Procedures here are essentially precompiled plans. An example might be "Focus radar system X on areas Y". An important issue is to recognize when and how to trigger procedural systems. NNs are one of AI technology that is able to deal with this recognition task. NNs model the brain in that given a stimulus a response is produced. Various given situations are input and the desired output to the individual situation is told to the NN. This way a network learns when approximately to employ specific procedures and to produce autonomously intelligent solutions. CBR systems store a library of cases, essentially plans or actions that are appropriate for a given situation. CBRs make analogy inferences. Other AI technologies that deal with uncertain and incomplete information are Bayesian Networks, Dempster-Schafer representation and Evidential Reasoning Systems. [6.10, 6.11 and 6.12]

to summarize the previous discussions, in providing a goal architecture for Computation-oriented applications for building the infrastructure of a C^3I system for elements should be considered:

- Formal specification requirements and analysis methods,
- System prototyping,
- International standards to aid interoperability,
- Implementation strategy.

There is no adequate foundation for a theory of C^3I and hence no guiding principles for system design and evolution.

In C2 systems decision making is not simply the selection of the perceived best option but the creation, evaluation and refinement of what the situation is and what can be done about it.

The majority of existing C2 systems are action information systems. They concentrate on the acquisition of information, on data processing and communication and on physical MMI aspects rather than on underlying analysis of the tactical decision processes themselves.

In the system requirements analysis the information from the following sources are incorporated:

- KB functions and tasks from the KB structure / functionality component,
- DB functions and tasks from the DB management system component,
- Interface modes and styles and explanation functions from the interface / displays component.
- Inference and reasoning goal and data-driven requirement, uncertainty handling algorithms from the inference / reasoning strategies component.

Three maxims for developing operational expert systems:

- Follow a problem-driven, not a technology-driven approach,
- Apply a structured and systematic development methodology to build operational expert systems,
- View the primary output of and R&D prototype expert system as detailed requirements analysis and functional specification, not as an operational system code.

Level of tasks for Intelligent C2 systems:

- Data acquisition from sensors,
- Sensor data interpretation and data fusion,
- Situations or thread assessment,
- Decision plans or goal generation for desired outcomes,
- Implementation of plans and monitoring of response / actions to ascertain success through sensor or intelligence.

At highest level, decision maker / commander has a set of a priori goals / plans. At highest level, due to continuously changing dynamic environment, new data / intelligence may effect the decision process by changing the goals to be achieved and the manner in which they are to be achieved.

Balance between full data assimilation with no planning solution and an incorrect solution based upon incomplete data is encountered.

C2 system must restrict the solution space if real time solutions are to be feasible, the risk of failing to generate any real time solutions suggests that the control system methodology accepts only partial goal satisfaction.

Two supplementary problems are:

- Recognizing that there exists an incomplete data,
- Overcoming incompleteness

The majority of AI systems have tended to be static information systems or systems which have dynamic networks to support their reasoning. Dynamic AI networks tend to utilize semantic nets, frames and blackboard systems. These systems support architecture's that are capable of supporting continuous data inputs and planning for an arbitrary range of problems.

Semantic nets and frames are particularly appropriate for detailed knowledge representations but, unlike blackboard systems are less able to deal with globally accessible areas for solution space.

The main activities of the higher levels of a C2 system are situation assessment and resource allocation / planning, for which the blackboard expert systems architecture has been advocated [6.5].

A blackboard architecture is based upon a group of experts or knowledge sources, who are ignorant of each others expertise. They surround a "blackboard" or global data area. As new data is written on the blackboard, each expert examines the blackboard to see if his expertise / knowledge has nay contribution to make through a new hypothesis. This in turn may be used by the other knowledge sources to jointly contribute to the problem solution. The blackboard is both a representation of the current state of knowledge and a means of communication among the experts.

The blackboard expert systems approach is clearly appropriate to C2 subsystems that involve a strong planning element.

Back chaining Bayesian Inference expert systems have attracted considerable interest in C2 systems such as IKBS to provide advice on electronic warfare plan evaluation.

AI provides the tools to utilize an experts (commanders) knowledge for effective use in a real-time computer based systems would act as a special purpose intelligent controller in the support of the commander.

6.4 REFERENCES

[6. 1] Özsu M.T., Valduriez P., "Principles of Distributed Database Systems", Prentice-Hall, Inc. 1991.

[6. 2] S. B. Navathe, and J-L. Weldon. "An Integrated Approach to Database Design. In Data Base Design Technologies I: Requirements and Logical Structures", Lecture Notes in Computer Science 132, New York: Springer-Verlag, 1982, pp. 1-30.

[6. 3] Davenport R. A. "Design of Distributed Data Base Systems". Comput. J. (1981), 24(1): 31-41.

[6. 4] Andriole S., Halpin S.M., "Information Technology for Command & Control: Methods and Tools for System Development & Evaluation", IEEE Press, 1991.

[6. 5] Harris, C. J., "Application of Artificial Intelligence to Command and Control Systems", Peter Perengrinus Ltd., 1987.

[6. 6] Ince, A. N., et al, "Planning and Architectural Design of Integrated Services Digital Networks" Kluwer Academic Publishers, Boston, 1995.

[6. 7] Mirchandani, S. et al, "FDDI Technology and Applications" John Wiley and Sons, 1993.

[6. 8] Albert, B, Jayasunan, A. P., "FDDI and FDD-II Architecture Protocols and Performance", Artech House, 1994.

[6. 9] Schwartz, M., "Telecommunication Networks: Protocols, Modeling and Analysis", Addison Wesley Pub. Co. 1987.

[6. 10] Harris, C. J., "Advances in Command, Control & Communication Systems", Peter Peregrinuo Ltd., 1987.

[6. 11] Demster, A. P.," A Generalisation of Bayesian Inference", Journal of Royal Statistical Society, Series B, 30 (2), 1968.

[6. 12] Shafer, G., "A Mathematical Theory of Evidence", Princeton University Press, 1976.

ANNEX 6-A

DATA TRAFFIC CALCULATIONS

1. INTRODUCTION

One of the most important system parameters which determine the system performance and the technology to be used to implement the system is the volume and speed of data flow between the users in a node (intranodal traffic) and between the nodes of the C^3I system (internodal traffic).

Data exchange requirements will be different for each user terminal depending on its command control activities. In addition it is necessary to take into account the traffic, such as the traffic generated by the servers, between the elements of the information system.

It is for these reasons that when dealing with traffic modelling and analysis it is convenient to group the user terminals into three categories from the view point of their traffic characteristics i.e., frequency of useage and the command control applications performed. In the sections below we shall give the number of user terminals and servers which may exist in a typical military headquarters which is assumed to consist of peace and war establishments, as well as their traffic characteristics followed by calculations of the maximum traffic generated by these terminals and servers.

2. ASSUMPTIONS

In what follows we shall try to estimate the average traffic load and network delays in a typical C^3I primary node using an analytical model based on assumptions which we shall make regarding the organization including its structure and the number of and the maximum traffic generated by the user terminals and servers connected to the nodes (see Fig. 6-A.1).

As a basis for these traffic calculations we shall postulate a possible node configuration as given in Fig 6A.1 which is considered a good representative configuration, certainly for a strategic military C^3I system. We shall perform the traffic calculations for operations in normal (peace) time when the terminals (with non-volatile memories) to be used will have their discs removed into a secure area for security reasons and under stressed conditions (crises/tension and war time) when the terminals incorperating their disc units will be used from the secure area. The following traffic sources are assumed for the calculations below:

3. TRAFFIC SOURCES

3.1 Normal (Peace) Time

3.1.1 Terminals (CMW UNIX Work Stations without discs)

- **3.1.1.1** RARP Packets (Reverse Address Resolution Protocol)
- **3.1.1.2** OS, File service for loading software
- **3.1.1.3** File service for swapping
- **3.1.1.4** Recall and precessing related to application menu and graphics windows

3.1.2 Distributed Data Base Traffic

- **3.1.2.1** Updating (within and outsite Hqs)
- **3.1.2.2** Replication of updates in alternate Hqs
- **3.1.2.3** Replication
- **3.1.2.4** Archiving

3.1.3 GIS Server

- **3.1.3.1** Data from DBMS servers

3.1.4 Servers for Briefing Application

- **3.1.4.1** Data from data bases
- **3.1.4.2** Data from GIS servers

3.2 Crises/Tension and War Situation

In this time period, terminals (UNIX WS) in the secure area (eg. war Headquarters) will be used. Since these terminals incorporate disc units, the traffic of the category 3.1.1.1, 3.1.1.2, and 3.1.1.3 above will be absent. The other traffic sources, however, will generate more intense traffic in this period. Some of the traffic sources such as those in 3.1.1.1 and 3.1.1.2 will disappear in the steady-state condition. We shall neglect the traffic created by E-mail and printing services.

4. MAXIMUM TRAFFIC CALCULATIONS

The figures given below for the number of traffic sources (Work Station) and for the traffic density for each source are regarded as maximum figures that will be used for traffic load calculation for intranodal traffic.

For simplicity sake, we shall assume that the priority scheduling mechanism normally used in the backbone network (FFDI) which apportions capacity to the servers and processors in proportion to the traffic they generate, is not operative.

4.1 Normal / Peace -Time Traffic

4.1.1 Terminals:

In a study conducted by the authors in an important major Hqs it was established that there could be about 300 work stations in the peace Hqs and 200 WS's in the wartime location.

4.1.1.1 RARP packets: The load introduced by the booting of a terminal to the network is 8 Kb/s. However, this load disappears in the steady-state condition.

4.1.1.2 Loading oriented File service (OS) creates 1.8 Mbit/20 s = 0.9 Mbit/s traffic at the opening. Although this load disappears when the system is operating, we shall assume that 10 % of the terminals will affect the system.

4.1.1.3 Swapping oriented file service introduces traffic by swapping (page-out) and loading new modules to memory (page-in) which are taken as equal. These operations are done over the network since the terminals in question do not incorporate disc units. We shall assume that there are terminals generating low (10 kbit/s), medium (30 kbit/s) and high (100 kbit/s) traffic density. The numbers of terminals in each category can be approximately determined from the frequency of the data flow in the decision/actions (Table 2.3) and from the characteristics of the operations performed (text, data, input, graphicsand imagery data). The studies conducted by the authors show that the proportion of the total number of terminals in each category varies with the functional area and, of the total terminals in the Hqs about 80 % of them generate low, 15 % medium and 5 % generate high density traffic.

4.1.2 Distributed Data Base Traffic

4.1.2.1 Updating (within the main Hqs and outside) :

1 Gbyte/day of traffic updating and entering new data into the data bases is assumed. This is considered to be a worst case assumption and the traffic load thus calculated would be much above the real average value for this type of traffic. Considering also the following traffic

i) Updating within the headquarters
ii) Replication of updating to alternative headquarters
iii) Archiving
iv) Updating of databases outside the headquarters

the traffic figure mentioned above may be quadrupled. If the operations in question are assumed to be carried out in six hours within the day average value of the distributed data base traffic can be taken to be about 1.4 Mbit/sec.

4.1.3 GIS Server

The map data of the GIS server will be 350 Mbyte/sec. With the assumption that this is updated and querried within the day the GIS server can be considered to create an average traffic load of 120 Kbit/s.

4.1.4 Briefing Application Server

The traffic load of the brifing application server is taken to be the sum of the loads of the data base and GIS servers and therefore is taken to be 1.50 Mbit/s at most.

4.2 Tension/Crises & Wartime Traffic

Making use of the information given in Sec. 3.2 above, the maximum possible traffic is estimated to be about 42 Mbit/s in this time period. It is to be noted that the emprical and assumed traffic values given in Secs. 3.1.1.1, 3.1.1.2 etc. are the maximum possible values.

5. CALCULATION OF AVERAGE TRAFFIC USING AN ANALYTICAL MODEL OF THE NETWORK

As shown in Chapter 5, for the type of configuration postulated here and for the traffic load estimated above, FDDI (Fiber Distributed Data Interface) would be a good condidate for use for the backbone network of the main Hqs and other primary nodes of the system [6.7] and [6.8].

FDDI protocols are based on Token Passing principle and this prevents any loss of data due to packet collisions. Nevertheless, for a network interface layer to pass on the data packets that it has in its queue it needs to have a token for which it will have to wait. This would entail a time loss which would affect the efficiency of the network. Efficiency is a measure of the percentage of the FFDI network bandwidth which can be used by the work stations connected to the network.

Efficiency would depend on the amount of data that would pass with each token and with the exchange of information between interface units for the purpose of increasing efficiency. However, the time loss mentioned above results in a delay for each packet which we can call Interface Layer Delay

The network efficiency and network interface delay in an FDDI-2 are given by the following expressions:

Efficiency = $n (T-L) / (nT+L)$

Network Interface Layer Delay = $(n-1) T+2L$

where n = Number of terminals accessing the backbone network
T = Token hold time
L = Ring delay (2.3 ns for 200 km)

As has been stated above the number of work stations in the system can be as high as 500. However, since these work stations and servers are assumed to be connected to FDDI-2 via "Dual Access Concentrators (DAC)", as shown in Fig. 6-A1, the number of terminals on the backbone can be calculated in accordance with the figures given in Table 6-A1 below:

Table 6-A1 Number of Terminals connected to the Backbone Network

Terminal Traffic Requirement	Connection Type	Total Number	Number of DAC's
Server	1-port DAC	13	13
High	8-port DAC	500×0.005=25	25/8 ≈ 4
Medium	16-port DAC	500×0.15=75	75/15=5
Low	32-port DAC	500×0.8=400	400/32 ≈14
		Total	**36**

The numbers of work stations given in the table 6-A.1 have been derived from the work station category figures in Sec. 4.1.1.3 above.

If T is taken to be typically 1 ms and the network cable to be 4 km in length, the efficiency and delay figures for this situation is given as:

Efficiency = 36(1 - 0.046)/(36 × 1 + 0.046) = 95%
Delay in Network Interface Unit = 36 + 0.046 = 36.046 ms = 0.036 s
(when queue is empty)

In order to compute average values it is necessary to form a queueing model of the FDDI backbone. This is shown in Fig. 6-A.2.

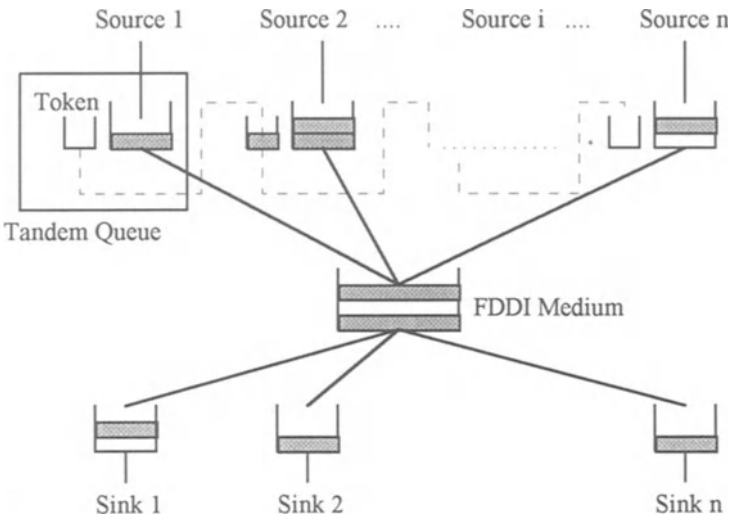


Figure 6-A.2 FDDI Queueing Model

The tandem queues shown on the upper part of the figure define FDDI protocols. In order for any source (source i) to transfer a message it must have a token in its token queue. Following the transmission of the message the token is transferred to the next token queue.

Let us assume that the message arrival distribution is of Poisson type:

$$P(R,t) = (\lambda t)^R e^{-\lambda t} / R!$$

which gives the number of message arrivals, R, in the time interval t.

When steady-state is reached, the probability density function of the arrival rate, can be represented by $f_\lambda(t)$ and the probability density function of the token rotation time, by $F_\tau(t)$. It follows from this that the probability of the processor being blocked is equal to the probability of arrival in the message queue of more than one message in one rotation period. This probability can be taken as $p = \lambda\bar{\tau}$ where $\bar{\tau}$ represents the expected value of τ.

With the simplifying assumptions that the message arrival rate λ is the same for all interfaces and that the processor are equidistant from each other then we can take the delay times due to token transfers to be the same for all interfaces. It is to be noted that in this model, with $\lambda = \lambda_{max}$ we get the maximum possible traffic in the backbone network. In this case the average rotation period can be shown [6.8] to be given by:

$$\bar{\tau} = \frac{NT_c}{1 - \lambda N(N+1)t_s / 3}$$

where N = Number of processors on the network and the network interfaces,
T_c = Delay time due to token transfer and
t_s = Message transfer time between two adjacent interfaces

from the expression above the value of $\bar{\tau}$ can be calculated to be about 25 ms. Here, λ is computed by dividing the maximum traffic by the number of processor (N = 18, assuming a half of the processors to be active at any given time on the average) and by the packet size and t_s is computed from the network length.

The average number, M_Q of processes waiting in the queue of any processor is given [6.9] by:

$$\bar{M}_Q = \left(\lambda\bar{\tau} + e^{-\lambda\bar{\tau}}\right) - 1$$

and the average delay time of a packet in a queue, E, is given by:

$$E = M_Q \times \bar{\tau}$$

from which the average delay time of a packet can be calculated to be about 34 ms. It is to be noted that the delay time in the queue is the time necessary for a packet to be processed in a network interface unit. The total delay time for this packet to be transferred to the network can now be computed by taking into account also the delay time (previously calculated to be about 36 ms) associated with the transfer of the token to the interface unit is 34 + 36 = 70 ms. This total delay gives approximately the delay caused by processing of the packet on the network and by the network traffic (assuming packet transmission time to be negligibly small).

6. NETWORK CAPACITY AND MAXIMUM TRAFFIC

The capacity of a FDDI-2 backbone can be taken to be 200 Mbit/s at present and the available capacity in this case is therefore found to be 200 × 0.95 = 190 Mbit/s. As mentioned previously this does not take into account the protocol overheads. In the FDDI protocols, 87 octets are assigned to the header and tail and 256 octets to the data part of the packet.

From these figures we can compute the protocol overhead as 25% and the maximum traffic as 42 + (42 × 0.25) = 83 Mb/s. The conclusion that can be drawn from the above calculation is that the expected maximum traffic (53 Mb/s) in the backbone network of a main HQ would be about a quarter of the capacity of a FDDI-2 cable which, if used, would accommodate three to four times more traffic than the maximum traffic that is foreseen now and would leave ample capacity for video requirements.

7. INQUIRY RESPONSE TIME

Processing delay of any process in a network consists of processing delay plus delay caused by the network itself. Delay due to the network was calculated in the previous section. The process time shows differences for different processes. In order to receive data unchanged it is necessary to receive it within a defined time limit. It is for this reason than Inquiry Response Time must be known and specified. This is defined as the length of time between an indication of the end of an inquiry and the display of the first character of the response at a user terminal. Inquiry response time is a statistical quantity. There are applications where it is a requirement that 90% of the queries must be responded within 3 seconds. When this figure of 3 sec. is compared with the network delay (0.07 sec.) we see that the latter is much smaller than the former and consequently the network delay can be said not to affect the system performance in the assumed network.

NATO standards for response times for some processes are listed in Table 6-A.2:

Table 6-A.2 Response Time For Some Processes

Process Type	Response Time
Updating of operational data bases: The time elapsed between demanding the update process and updating of one record	14 sec.
Identification of user login: The time elapsed between attempting to login by entering the user name/password and accepting/denying the user aften identification (this must be tested when there are one less than the maximum number of users)	60 s
Display of text: The time elapsed between demanding and displaying a screen-full of text	5 s
Display of graphics: The time elapsed between demanding and displaying a screen-full of graphics	30 s
Updating of data bases by message: The time elapsed between receiving the last character of a message (EOM) and updating the data base and archiving the message	46 s
Display of window: The time elapsed between demanding and the display of user interface unit (window, menu)	3 s

The values given above are used to define the characteristics of the hardware to be used for data base, briefing, GIS and message handling applications so that the values in Table 6-A.2 are not exceeded. Standards for process times corresponding to different processing loads are used to define the server architecture and capacities (number of processors,processor speeds etc.) so as to comply with these

standards. The applications mentioned above being mostly input/output dependent (e. g. data base applications) the data transfer rate itself in fact defines the performance of the system. For example, in a query involving sequential read and assuming that the transfer rate of a SCSI-II disc is 10 Mbit/s and the access rate of SIMM memory modules (volatile, fast, random access memory) is 70 ns the processing time for 50 Mhz processor receiving from memory data at a rate of 1.25 Mbyte/s would be 3 millisecond.

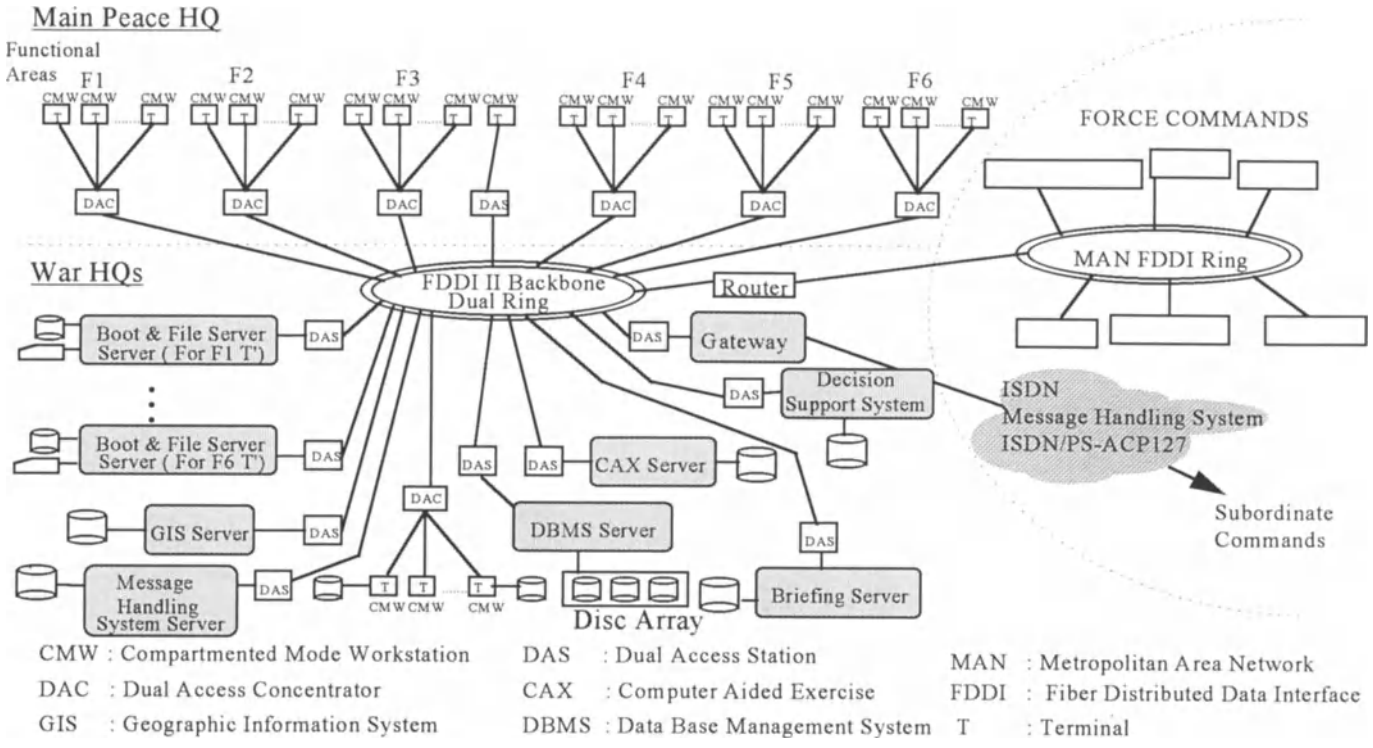


Figure 6-A.1 Configuration Block Diagram for a Primary Node and its Connections to Other Primary and Secondary Nodes

CHAPTER 7

SECURITY ARCHITECTURE

7.1. TRUST IN THE C^3I SYSTEM

As technological advances make information more available and more vulnerable, awareness of information security issues such as personal privacy, computer fraud and legislation, and national security related concerns are brought to light. A meaningful information security program depends on the support from the top management of the organization. It is the responsibility of the Information Security Administrator to present management with a clear view of the threats and alternative solutions for countering those threats. Management must weigh potential losses versus the cost of limiting the exposures to make a decision either to accept or reduce the organization's risk potential, and direct corrective action. The final success of an organization is very much in danger, if the top management of the organization does not understand the objectives of the security program.

For a computer system to be secure it must provide specific mechanisms for protecting itself against threats to the confidentiality, integrity and availability of the information and/or resources that it stores, processes or owns. Obviously this protection must be balanced with the threats to the information and/or resources.

As information systems have become widely used, users have become more aware that their systems need to function as required and react sensibly to accidental or malicious misuse. It has been widely accepted that information systems should be safe, dependable and secure. The security of C^3I systems requires significant attention, because they are used as decision support system handling information that is vital to the organization. Therefore, C^3I systems are very attractive as targets for potential opponents.

Trust in the C^3I system is a function of the confidentiality, integrity, authenticity and authorization properties.

7.1.1 Confidentiality

Confidentiality means that the contents of the information processed, stored or transmitted are protected. This means that the existence of a piece of information, its sources, its recipients,its content, the meaning or structure of system resources dealing with this piece of information must not be made available or disclosed in any way to unauthorized entities.

7.1.2 Integrity

Integrity ensures that there is some form of proof that information has not been modified either accidentally or deliberately by non-authorised entities. Integrity ensures that all resources in the system are always available in a "high quality" form, i.e., that the content, meaning, structure and function of each resource are always available and preserved in its correct form. It should be noted that the definition of integrity also implies availability.

7.1.3 Authenticity

This is the property that the identity of each element of the system may be unambiguously established and validated, i.e., that each element of the system is always the one claimed or required.

7.1.4 Authorization

Authorization is the property that all actions in the system are always initiated, executed and terminated in accordance with requirements and constraints of the security policy.

7.2. THREATS TO COMPUTER SYSTEMS

A threat to a computer system is defined as any unauthorized action that could adversely affect computer system assets. The primary threats include unauthorized disclosure, change,blocking, or theft of assets. Identifying threats to computer and network systems is a requirements analysis activity.

Threat identification approaches follow an ad hoc, unstructured process. To offer a rigorous and repeatable method of threat identification for software and system engineering efforts, the use of a hierarchical technique known as a threat tree is proposed in [7.9]. In particular, the threat statements from various standards documents (e.g. ISO 7498-2, X.400, ECMA TR/46, DAF Security) should be taken into account. Some important threat types are explained in the following [7.2].

Constructing a threat tree is an important first step in System Security Engineering (SSE) because it points to the types of damage that system assets can incur. Proposing security safeguards before threats are determined wastes money, causes unnecessary annoyance to system users, and adds little protection for critical assets.

Masquerading: This simply means the pretense of one entity to be another entity. By masquerading, an entity can get hold of privileges which it is not authorized to have. This can happen at different levels in a distributed system: within a computer system, a user or process might masquerade as another to gain access to a file or memory to which it is not authorized, while over a network, a masquerading user or host may deceive the receiver about its real identity.

Unauthorized use of Resources: This includes unauthorized access to both resources on the networks as well as within a system. For instance, within a computer system, this threat corresponds to users or processes accessing files, memory or processor without authorization. Over a network, the threat may be in the form of accessing a network resource. This may be a network component such as a printer or a terminal, or a more complex one such as a database, or some applications within the database. Thus unauthorized use of resources may lead to theft of computing and communications resources, or to the unauthorized destruction, modification, disclosure of information related to the business.

Unauthorized Disclosure and Illegal Flow of Information: This threat involves unauthorized disclosure and illegal flow of information stored, processed or transferred in a distributed system, both internal and external to the user organizations. Within a system, such an attack may occur in the form of unauthorized reading of stored information, while over the network, the means of attack might be wiretapping or traffic analysis.

Unauthorized Alteration of Resources on Information: Unauthorized alteration of information may occur both within a system (by writing into memory) and over the network (through active wire-tapping). The latter attack may be used in combination with other attacks such as replay whereby a message or part of a message is repeated intentionally to produce an illegal effect. This threat may also involve unauthorized introduction (removal) of resources into (from) a distributed system.

Repudiation of Actions: This is a threat against accountability in organizations. For instance, a repudiation attack can occur whereby the sender (or the receiver) of a message denies having sent (or received) the information. For instance, a customer engages in a transaction with a bank to debit a certain amount from his account, but later denies having sent the message. A similar attack can occur at the receiving end; for instance, a firm denying the receipt of a particular bid offer for the tender even though it actually did receive that offer.

Unauthorized Denial of Service: Here, the attacker acts to deny resources or services to entities which are authorized to use them. For instance, within a computer system an entity may lock a file hereby denying access to other authorized entities. In the case of the network, the attack may involve blocking the access to the network by continuos deletion or generation of messages so that the target is either depleted or saturated with meaningless messages.

7.3. VULNERABILITIES IN COMPUTER SYSTEMS

A vulnerability is defined as a system characteristic (usually a feature or a flow) that may be exploited by a malicious intruder to cause a security threat.

Specific vulnerability views are discussed in the following subsections.

7.3.1 Personal View

This area of vulnerability analysis is often neglected, especially by experts who tend to focus on technical vulnerabilities. Disgruntled employees, for example, are a primary vulnerability if they are in a position to damage a given system.

7.3.2 Physical View

Another often-neglected area, the physical view includes all facility and site-oriented concerns, such as protecting computing resources from physical and environmental damage.

7.3.3 Operational View

This area involves the disclosure of information that results from the procedures or operations used by certain individuals within an organization. It ranges from disclosing a password by means of neglect to more subtle operational issues, such as allowing key personnel to gather in an observable or public location.

7.3.4 Communications View

In this view, vulnerabilities are examined in the context of the specific transmission media and associated devices that pass information between remote locations.

7.3.5 Network View

Most vulnerabilities associated with networks can be traced to shortcomings in network components, such as routers, hubs and bridges that do not support desirable access control and audit collection features.

7.3.6 Computing View

This view focuses on the vulnerabilities that may be present in the operating system and associated application infrastructure.

7.3.7 Information View

This comprises a system view that examines vulnerabilities based on the manner in which information is created, disseminated, stored and transferred.

7.4. ATTACKS ON COMPUTER SYSTEMS

Security attacks on computer systems are defined as heuristic procedures that exploit vulnerabilities in a manner that causes a threat.

Creating properly engineered, general taxonomies of attacks is not easy, because the details of a particular attack will always be system specific. Obtaining the most complete set of attacks on a given system will require at least the following two activities.

- A detailed comparison of the system environment, with taxonomies of attacks that have occurred previously in similar environments. This helps ensure that known attacks are avoided.

- A detailed investigation and generation analysis by individuals who have intimate knowledge of the target system. This provides added confidence that the system does not allow obvious attacks.

7.5. ESTIMATION OF SECURITY RISK

Once the process has identified the set of threats, vulnerabilities, and attacks for a given system, the system security engineer must consider the relationship between the various threats and their relative priorities. In the absence of a well-defined process, the system security engineer would be forced to determine, without guidance, how to alleviate an unprioritized list of security problems. This issue would be less critical if every system manager had sufficient resources and funds to mitigate every security problem completely. Even in such cases, however, inevitable tradeoffs would have to be considered between security and other important system attributes, such as usability.

To address this issue, the process incorporates an analytical step in which the security risk associated with all threats, vulnerabilities, and attacks is estimated using a mathematical model that incorporates the following factors:

- **Level of adversary effort (LAE):** How easily might a malicious intruder enact a given threat, access a given vulnerability, or complete a given attack? To estimate the degree of ease associated with a given threat, vulnerability, or attack, the process examines the number of possible attacks that can be used to enact that threat or exploit that vulnerability. This estimate is referred to as the level of adversary effort (LAE). Roughly, the LAE increases in proportion to how difficult it is for an intruder to attack a system.

- **System-Weighted Penalty:** The relative criticality associated with the assets affected by the threat, vulnerability, or attack must also be estimated. Generally, this estimate is based on how much impact the removal of that asset would have on the successful performance of the system mission, and is further adjusted according to the relative criticality of that mission. Assets associated with life-critical applications, for example, would generally be viewed as more

critical than comparable assets associated with computer games. The resultant criticality estimate is referred to in the process as the system-weighted penalty (SWP).

- **Security Risk:** The security risk of a particular threat or vulnerability is calculated from the LAE and SWP estimates using the empirical formula:

$$\text{Risk} = \text{SWP}^2 \div \text{LAE}$$

As SWP increases (i.e., the associated asset is more critical to the system), its security risk increases. Similarly, as LAE increases (i.e., it becomes more difficult for a malicious intruder to cause security problems), the security risk decreases. In the formula used, SWP is squared. Some system analysts, using an empirical process based on analysis of attacks and faults claim this relation as having the desired effect. Using this formula on a variety of SSE efforts has confirmed the validity of the relation.

Associating risks with all threats, vulnerabilities, and attacks offers a means of prioritizing the security problems for a given system. As one might expect, it is better to start with the most serious problem in any effort to improve the security of a computer or network system.

Some useful heuristics that should help the system security engineer to prioritize threats, vulnerabilities, and attacks include:

- Prioritizing threats only may relegate all vulnerabilities and attacks to the same priority. Therefore, high-priority threats should be viewed in relation to the vulnerabilities and attacks that caused them.

- Prioritizing vulnerabilities results in a system-architecture-oriented approach to security mitigation. This has been found useful for proposed system architectures that are being analyzed for security enhancement.

- Prioritizing attacks results in a hierarchy of security issues that will have varying effects on the system. This approach is often used for stable architectures that are not being considered for security retrofit (other than finding quick fixes for serious problems).

- Prioritizing everything, the most desirable approach, is also the most time-consuming and prompts the most questions. In vulnerabilities associated with multiple threats, for example, do we assign the maximum of the associated threat priorities, or some new priority that reflects its association with multiple threats? A similar question arises for attacks and associated vulnerabilities. The answers to these questions are typically generated based on engineering consensus and the unique characteristics of the system being analyzed.

7.6. SECURITY SOLUTIONS

Once a prioritized collection of security problems- either in the form of threats, vulnerabilities, attacks or some combination of these - has been identified for a given system environment or application, security solutions must be identified to mitigate the associated risks. Such mitigating ranges from complete avoidance of a particular problem to a slight alleviation of the effects of that problem. One difficulty in identifying security solutions, discussed later, is how difficult it is to accurately determine the exact mitigation effect of a particular solution. Certainly, any number of specific security solutions can be identified for all types of security problems that might arise on computer and network systems. Security solutions can be divided into two primary types: security safeguards and countermeasures. Security safeguards are roughly those solutions designed into a system in anticipation of a potential problem, whereas security countermeasures are responses to a detected or suspected security problem. This distinction is often blurred by many security engineers, but it is useful for explaining the SSE process to customers.

Security solutions can also be divided into those realized using automated security mechanisms, and those realized using non-automated operational security procedures. For example, a database could be protected by automated access control mechanisms, or by non-automated procedural guidelines (e.g., company policies).

Although it is not possible to describe all types of security safeguards and countermeasures available, some of the more common and useful approaches include:

7.6.1 Security Policy

The collection of security requirements incorporated into a computer or network system is generally referred to as a security policy. To support security policy, the computer security community has promoted the notion of a reference monitor, an abstract model of the mediation provided by a secure system against potential attacks. A security policy thus provides a detailed formal or informal specification of the reference monitor requirements for a system. Some system security engineers have found that customers rarely have well-defined security policies for their computing and networking environments, and would benefit from any effort to resolve this problem.

7.6.2 Authentication

Typically, before computing or networking services are granted to a requesting user, most systems require specific information from the user to verify his or her identity. Because this information should not be trusted (a malicious intruder might try to lie about his or her identity), systems generally require proof that this identity is accurate. The authentication function is used to ensure that reported identities are correct. Simply stated, authentication of a user must be established as a

combination of something he has (e. g. a smart-card) and something he knows (e. g. a pin-code).

The most frequently used approach involves simple passwords, but more sophisticated schemes exist, including the use of personal identification number (PIN), protected smart card or hand-held authenticators. In such schemes, it is common for the system to provide a challenge (usually a string of characters) to the user, who then must respond to the challenge as input to hand-held device. The device calculates the value of some predetermined computation using the challenge and returns the results to the system, which also performs the same computation. If the results match, then the system concludes that the user is authentic (or that an attacker has stolen the hand-held device).

System security engineers commonly work with customers to ensure that the optimal authentication approach is being used, especially in environments with many network entry points.

7.6.3 Access Control

Earlier, we mentioned the reference monitor model to help describe the mediation that secure systems often provide. This model is implemented by two types of access control: discretionary access control (DAC), in which users have the ability to control who can and cannot access their information and resources; and mandatory access control (MAC), in which use do not have this ability.

The most common DAC scheme encountered during an SSE analysis is the familiar UNIX permissions vector, which enables users to control, read, write and execute rights to their files and directories. A similar DAC scheme that is becoming more common is found in systems such as Microsoft's Windows NT.

An Access Control List (ACL) is a list attached to every object to be protected. It defines who can perform the available types of access (e.g., read, write, execute) on that object. The main benefit of DAC schemes, their flexibility, can also be their main detriment. Because users control the settings, Trojan horse programs could direct users to inadvertently change these settings to some undesirable configuration.

In the most common MAC scheme, internal security label representations are included on a system for all information repositories (e.g., files and directories) and active invoking agents (e.g., processes). A security policy then defines the mediation requirements to be enforced based on these security labels. For example, processes with low-security labels might not be allowed to read files with high-security labels. This type of mediation often requires operating system enhancements or explicit kernel functionality.

Because MAC is controlled by system or security administrators, its normal users cannot be fooled by Trojan horse attacks into changing MAC settings. It is therefore a desirable choice in many SSE efforts for protecting highly sensitive information. It is also, however, less flexible than DAC, and users have complained

about the distinct reduction in system usability that often accompanies migration to a MAC scheme.

7.6.4 Auditing and Intrusion Detection

The security safeguards described earlier are all considered intrusive - they affect user operations directly. A less intrusive technology available to the system security engineer involves on-line auditing of all security-relevant activity into a protected log for later perusal. This deters intruders who do not want to get caught (as opposed to so-called kamikaze, or truck bomb, attacks), but its non-intrusive aspect makes it a poor choice for avoiding potentially catastrophic security problems.

In addition, auditing schemes may produce such voluminous quantities of information (e.g., often several megabytes of information on a daily basis) that system and security administrators would be hard - pressed to go through this data manually in a reasonable time frame. As a result, automated data reduction, data summary, and intrusion-detection approaches and tools have been developed to scan, process, and interpret audit data.

The most common intrusion - detection algorithm involves capturing expected behavior in an internal representation that is often referred to as a profile. Profiles are created based on the best available predictions of system or user behavior. Then, audit data, which represent what was observed, can be filtered into a similar internal representation so that they can be compared with the associated profile. If expected behavior differs from observed behavior, then a problem might exist, precipitating alarms or other warnings. Additional details on such schemes are available in many works.

7.6.5 Trusted Computing Base (TCB)

The software (and hardware) mechanisms that implement the security safeguards described earlier must be analyzed and validated to operate properly; if they are not, the security aspects of a system would have to be viewed as suspect. System security engineers have therefore developed what they call a trusted computing base (TCB), which localizes all security safeguard implementations for a given system. A TCB is designed to minimize complexity, ensure tamper-proof operation, and avoid bypass approaches. In networked environments, identifying a network TCB (or NTCB) is complicated by the complexities typical of a distributed architecture.

7.6.6 Network Security

All the safeguard solutions discussed earlier refer to any type of application, including networking, database, embedded, and real-time. Nevertheless, because networking has become such an important application, especially in the light of the explosive growth of the Internet, certain security safeguards have arisen that are most commonly found in network systems. In particular, encryption is usually included in network applications that require secrecy in communications. By

encrypting transmissions, users reduce the likelihood that attackers can listen to their data or voice conversations.

Ingenious algorithms have also been developed to maximize the usefulness of encryption. By applying a key (or parameterization) to the encryption function, users can set up single-key transmissions in which both the sender and receiver of information share a key that is hidden to all others. This procedure ensures secrecy and authentication (i.e., the sender must have been authentic if the proper key was used to encrypt the transmission).

In single-key solutions, however, complex networking environments, such as the Internet, require too many keys. As a result, two-key transmissions exist in which a public key, known to all, and a private key, known only to its owner, are required to encrypt and decrypt a message. Thus, senders can encrypt with the public key of the recipient to ensure secrecy, because only the receiver can decrypt that message. Similarly, senders can encrypt with their private key to ensure authentication, because no other entity owns that key.

7.6.7 Integrating Security Solutions

Although identifying candidate security solutions is a critical step in SSE process, it is usually not until the security solution integration step has been initiated that progress is visible in a given system, environment, or application. This vital step in the process determines an optimal selection and integration of the best available security technology into the existing target system. As a system is being conceptualized, the integration is targeted at the system requirements. If the system already exists, however, and its security properties are to be analyzed, the integration must be aimed at the existing characteristics and attributes of that system.

Because this integration mainly involves system-specific considerations, it is difficult to generalize about the exact placement of security solutions in new or existing systems. In making these integration decisions, however, the system security engineer should be guided by five particularly significant factors - criticality, cost, threat, impact, and migration.

Criticality. The most critical components of an architecture should generally be viewed as higher-priority candidates for safeguard integration than lower-priority components.

Cost. Safeguard integration is an iterative process that terminates when all security threats have been suitably mitigated, or when all available funds and resources have been spent.

Threat. Safeguards must be selected to optimally counter known threats. Different safeguards may have different effects on a threat. For example, mandatory access control and auditing are both preventive mechanisms, but in slightly different ways. Mandatory Access Control mediates requests using explicit functional mechanisms,

whereas auditing mediates requests indirectly by increasing the likelihood that an intruder will get caught.

Impact. Security safeguards should be expected to have some impact on certain aspects of a system or environment. The most obvious area of such an impact is the usability attribute. Security features must be selected to minimize unnecessary and unwelcome effects on any critical environment.

Migration. The last principle of security safeguard integration is that provision for future migration of both security and nonsecurity features must be included in the integration strategy.

7.7. IMPLEMENTATION OF SECURE SYSTEMS BASED ON TCSEC

Security plays a vital role in the design, development and practical use of the distributed computing environment, for greater availability and access to information which in turn imply that distributed systems may be more prone to attacks.

In addressing overall security in a distributed system, it is necessary to integrate computer system security and communication security measures to protect information both within the system and between the systems. [7.2]

An attempt to provide a standard framework in which the total security environment for a system may be examined in an organized way can be listed as follows [7.3]:

- System Mission: A statement describing the purpose and function of the system, the sensitivity and criticality of its data and processes, and the security requirements and environment of the system.
- Security Policy: A written interpretation of all existing laws, policies, regulations, and guidance as they apply to the security of the system, its processes, data and products.
- Security Organizational Structure: A formalized hierarchy of specialized security management positions, each having detailed responsibilities, clearly defined authorities, and appropriate span of control.
- Security Implementation Procedures: A compilation of local regulations, operational plans and procedures, which, if properly implemented will ensure compliance with the stated security policy.
- Security Education, Training and Awareness: A formal program that ensures all security management personnel are trained in their respective disciplines and all system users are aware of their security responsibilities.
- Physical and Environmental System Protection: The facility characteristics and operational procedures used to control physical access to the system, its processes, data and products and to protect the system from environment hazards.
- System Connectivity Controls: The communications architecture and topology designed to control electronic linkage to the system.
- System Access Controls: All identification and authentication control mechanisms used to control logical access to the system, its processes, data and products.
- System Administration Controls: All actions taken to ensure optimal use and integrity of system security features and security hardware / software.
- Storage Media Controls: All actions taken and resources available to control the access to, protect the integrity of, ensure the availability of, and the proper disposal of storage media associated with the operation of the system.
- Accountability Controls: All actions and resources that consistently collect, record, trace and resolve all actions that have security implications.
- Assurance: The sum of all actions and resources that provide a degree of trustworthiness and credibility to all aspects of system operations.

After presenting these aspects of examining security, we will consider their realization given in the Trusted Computer System Evaluation Criteria (TCSEC) commonly known as Orange Book [7.7].

7.7.1 Evaluation of Secure Systems

The Orange Book defines four broad hierarchical divisions of security protection [7.1]. In increasing order of trust, they are:

D Minimal security
C Discretionary protection
B Mandatory protection
A Verified protection

Each division consists of one or more numbered classes, with higher numbers indicating a greater degree of security. For example, division C contains two distinct classes (C2 offers more security than C1); division B contains three classes (B3 offers more security than B2, which offers more security than B1); division A currently contains only one class.

The National Computer Security Center (NCSC) is responsible in the US goverment environment for evaluating secure computer systems and related security products (for example, network security products and add-on products such as fingerprint scanners and electronic access control tokens). The NCSC uses the security requirements described in the Orange Book as a guide in evaluating security products through its Trusted Product Evaluation Program (TPEP).

7.7.2 Security Policy Requirements

Security policy requirements [7.1] are the first set of security requirements defined in the Orange Book. As discussed earlier, a security policy states the rules enforced by the system to provide the necessary degree of security. The Orange Book defines the following specific requirements in the security policy category:

- Discretionary access control.
- Object reuse.
- Labels.
- Mandatory access control.

7.7.2.1 Discretionary Access Control

In contrast to mandatory access control, in which the system controls access, DAC is applied at the discretion of the user or a program executing on behalf of the user. With DAC, you can choose to give away your data; with MAC, you can't. Discretionary Access Control (DAC) Requirements for security classes are given in Table 7.1.

Table 7.1 . Discretionary Access Control (DAC) Requirements

Class	Requirement
C1	The system does not need to distinguish between individual users, only between those who are allowed to access a file and those who are not; for example, access to a file might be restricted to members of the WORK group. At this level, the system also does not need to distinguish between types of access (e.g., read versus write). At this level, a user does not need to own a file (or have any other special privileges) in order to give the file away. The system need not protect newly created files or other objects.
C2, B1, B2	The system must be able to distinguish between individual users. The Orange Book uses the phrase, "These access controls shall be capable of including or excluding access to the granularity of a single user." At this level, a user must have some privilege or permission in order to give the file away; most implementations require that the user be the owner of the file. The system must protect newly created files or other objects.
B3, A1	The system must be able to distinguish between types of access (e.g., read versus write). Access control lists (ACLs) are specifically required. The system must also be able to indicate that a particular user is not allowed access. (Many evaluated systems with lower ratings support this feature as well). In addition, groups must be defined.

7.7.2.2 Object reuse

Object reuse requirements protect files, memory, and other objects in a trusted system from being accidentally accessed by users who aren't authorized to access them. A system's ordinary access control features determine who can and cannot access files, devices, and other objects that have been assigned to specific users. Object reuse requirements address what happens when these objects are reassigned. Object reuse requirements apply to systems rated at C2 and above. Common object reuse features implemented in trusted systems include:

- Clearing memory blocks or pages before they are allocated to a program or data.
- Clearing disk blocks when a file is scratched or before the blocks are allocated to a file.
- Degaussing magnetic tapes when they're no longer needed.
- Clearing window objects before they are reassigned to another user.
- Erasing password buffers after encryption.
- Clearing buffered pages, documents, or screens from the local memory of a terminal or printer.

7.7.2.3 Labels

Labels and mandatory access control are separate security policy requirements, but they work together. Beginning at the B1 level, the Orange Book requires that every subject (e.g., user, process) and storage object (e.g., file, directory, window, socket) have a sensitivity label associated with it. For B2 systems and above, all system resources (e.g., devices, ROM) must have sensitivity labels. A user's sensitivity label specifies the sensitivity level, or level of trust, associated with that user; a user's sensitivity label is usually called a clearance. A file's sensitivity label specifies the level of trust that a user must have to be able to access that file. The security requirements of this category are explained in the following subsections.

Label Integrity

Label integrity ensures that the sensitivity labels associated with subjects and objects are accurate representations of the security levels of these subjects and objects. Thus, a file sensitivity label, such as:

TOP SECRET [VENUS]

must actually be associated with a TOP SECRET file containing information about the planet Venus.

Exportation of Labeled Information

A trusted system must be sure that when information is written by the system, that information continues to have a protection mechanism associated with it. Two important ways of securing exported information are to assign security levels to output devices, and to write sensitivity labels along with data. Systems rated at B1 and above must provide secure export facilities.

Subject Sensitivity Labels

The subject sensitivity label requirements state that the system must notify a terminal user of any change in the security level associated with that user during an interactive session. This requirement applies to system rated at B2 and above.

Device Labels

The device labels requirements states that each physical device attached to your system must have minimum and maximum security levels associated with them, and that these levels are to be used to "enforce constraints imposed by the physical environments in which the devices are located".

7.7.2.4 Mandatory Access Control

Mandatory access control (MAC) is the final security policy requirement. Unlike discretionary access control (DAC), which allows users to specify, at their own

discretion, who can and cannot share their files, mandatory access control puts all such access decisions under the control of the system.

7.7.3 Accountability Requirements

Accountability requirements are the second set of security requirements defined in the Orange Book.

The accountability requirements specified in the Orange Book are:

- Identification and authentication.
- Trusted path.
- Audit.

7.7.3.1 Identification and Authentication

Identification and authentication (I&A) is a requirement at all levels of system security. The Orange Book requires the user to identify himself to the system before performing any work that will require interaction with Trusted Computing Base (TCB). I&A requirements for security classes are given in Table 7.2.

Table 7.2 Identification and Authentication (I&A) Requirements

Class	Requirement
C1	The system distinguishes only between users and unauthorized users; authorized users log into the system in some fashion, but do not need to have individual login IDs.
C2 and above	At all levels above C1, the Orange Book requires individual accountability. Each user has an individual ID. The login ID must be a unique identifier, not a company, department, group, or terminal ID. Once you've successfully logged in, the system uses your ID, and the security profile associated with it, to make access decisions. For example, if you attempt to modify a sensitive file, the system is able to check your authenticated user ID against the list of ID's representing users who are authorized to view and modify the data in that file. Only if your ID appears in that list will the system allow you to access the file. The system also uses your ID to audit your actions - that is, to keep track of what you're doing in the system. If a person or user repeatedly tries to access files he's not authorized to view, the system will know! The section "Audit" describes how this tracking takes place.

7.7.3.2 Trusted Path

A trusted path provides an unmistakable means by which a user (typically at a terminal or a workstation) can communicate directly with the TCB without having to interact with the system through untrusted (and possibly untrustworthy)

applications and layers of the operating system. A trusted path is a requirement for systems rated at B2 and above.

The simplest approach to implementing a trusted path is also the most expensive and the most unwieldy. Supply each user with two terminals - one is hardwired to the TCB and is used for communication with TCB (for certain security-critical functions such as logging in, changing sensitivity labels and security levels, etc.); the other is used for ordinary work. Because this isn't a very practical approach, most trusted paths involve setting up a particular key sequence on a terminal that signals the TCB to halt any (possibly untrustworthy) process that is running on the terminal and to establish the direct link to the TCB. The key sequence must be one that's used only to invoke secure software, and one that untrusted software can't intercept or spoof.

7.7.3.3 Audit

Auditing is the recording, examining, and reviewing of security-related activities in a trusted system. A security-related activity is any activity that relates to a subject's access of an object. In audit terms, such activities are often called events, and auditing itself is sometimes called event logging. Typical events include:

- Logins (successful or unsuccessful).
- Logouts.
- Remote system accesses.
- File opens, closes, renames, and deletions.
- Changes in privileges or security attributes (e.g., a change in a file's sensitivity label or a user's clearance).

Beginning at the C2 level, the Orange Book requires auditing. Auditing lets you perform two very useful security functions: surveillance and reconstruction. Surveillance is the monitoring of user activity. This type of auditing might prevent security violations from occurring, if only because users know they're being observed. Reconstruction is the ability to put together, in the event of a security violation, a record of what happened, what needs to be fixed, and who's responsible. Auditing Requirements are given in Table 7.3.

7.7.4 Assurance Requirements

The Orange Book specifies two types of assurance: operational assurance and life-cycle assurance. The operational assurance requirements specified in the Orange Book are:

- System architecture
- System Integrity
- Covert channel analysis
- Trusted facility management
- Trusted recovery.

Table 7.3 Auditing Requirements

Class	Requirement
C2	The system must audit security-related events and must protect audit data. A related identification and authentication requirement states that the system must associate a user's identity with all auditable actions taken by that user. The system must be able to audit selectively by user.
B1	The system must be able to audit any changes in security levels. Any security overrides must be audited. The system must be able to audit selectively by security level (e.g., all users with TOP SECRET clearances).
B2	The system must audit events that might be used to exploit covert storage channels (See the section "Covert Channel Analysis" later in this chapter).
B3, A1	The system must be able to monitor the accumulation of security events that may indicate an "imminent violation of security policy". The mechanism must notify the system administration in some fashion - example, by sounding an alarm or sending a particular type of message - when certain predefined thresholds are exceeded, and must take what the Orange Book calls "the latest disruptive action to terminate the event". This might involve logging an offending user off the system.

Life-cycle assurance features ensure that a trusted system is designed, developed, and maintained with formal and rigidly controlled standards. The life-cycle assurance requirements are:

- Security testing
- Design specification and verification
- Configuration management
- Trusted distribution.

7.7.4.1 System Architecture

The system architecture requirement has to do with the way a system is designed to make security possible - if not inevitable.
Although systems at the lower levels (C1 through B1 or even B2) need not be designed specifically for security, they must support sound principles of hardware and operating system design, as well as the ability to support specific security features that might be added to these systems. System Architecture requirements are given in Table 7.4.

Table 7.4 System Architecture

Class	Requirement
All classes	A protected execution domain for security-relevant functions (protected from both deliberate tampering and inadvertent modification). For example, this means that privileged programs (such as those that audit security events) can't be interfered with by user programs. The Orange Book dictates that the TCB must maintain its own domain, free from external interference or tampering. Some systems have a clear distinction between user and system areas. Other systems have more complex, ring-based architectures in which there may be as many as ten distinct, increasingly more privileged domains. In a typical ring-based architecture, the TCB, or security kernel, occupies the innermost ring; user programs occupy the outermost ring; in between are such intermediate processes as operating system services and administration.
All classes	Protection of resources so they're subject to access control and auditing. This includes the protection of obvious security-critical resources - for example, such techniques as putting an access control list on the password file - as well as the protection of user files so they won't be erroneously or deliberately accessed by other users.
B1 and above	Process isolation through distinct address spaces. This ensures that when multiple processes run concurrently, they won't interfere with each other - by accident or design - by writing each other's memory, changing each other's instructions, and so on. It also ensures that the system can keep track of everything it needs (e.g., registers, status information) to switch from one process to another, an essential element of a multi-processing system. Virtual memory techniques are often used to keep multiple processes each in their own virtual address spaces without interfering with, or even being aware of, each other.
B2 and above	TCB modularity and the enforcement of least privilege in the design of these modules. Least privilege has a number of meanings in a trusted system. In terms of system architecture, it means that processes have no more privilege than they need to perform their functions. Only those modules that really need complete system privileges are to be located in the security kernel (e.g., in the innermost ring). Other, less critical, modules should call on more privileged routines only as needed and only for the duration of the needed operation.
B2 and above	Hardware features such as segmentation. Segmentation is a hardware protection feature. Systems supporting segmentation divide their virtual memory into segments. A process occupies calls, segments are of several types. System processes can access all types; user processes cannot access segments that are restricted to system use. Certain instructions are restricted and can be issued

	only by privileged system processes. The result is that unprivileged user processes cannot access or modify the memory used by the operating system.
B3 and above	A precise and simple protection mechanism that enforces such features as layering, abstraction, and data hiding. Systems of this kind have a very structured, hierarchical design in which system functions are layered. The lower layers of the hierarchy perform certain basic functions; the higher layers may perform more complex functions. Layers communicate with each other through calls via clearly defined interfaces. Data hiding means that a layer in the hierarchy has no access to data outside itself; data handled by other layers is hidden.

7.7.4.2 System Integrity

System integrity means that the hardware and firmware must work and must be tested to ensure that they keep working. For all levels, the Orange Book states that "Hardware and software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB".

7.7.4.3 Covert Channel Analysis

A covert channel is an information path that's not ordinarily used for communication in a system and therefore isn't protected by the system's normal security mechanism. Covert channels have a nice subversive sound to them. They're a secret way to convey information to another person or program. Covert Channel Requirements are given in Table 7.5.

Table 7.5 Covert Channel Requirements

Class	Requirement
B2	The system must protect against covert storage channels. System developers must perform a "covert channel analysis" (a thorough search), often using information flow methods, for all covert storage channels.
B3, A1	The system must protect against covert storage channels and covert timing channels. System developers must perform a covert channel analysis for both types of covert channels.

7.7.4.4 Trusted Facility Management

Trusted facility management is the assignment of a specific individual to administer the security-related functions of a system. It's somewhat surprising that trusted facility management is an assurance requirement only for highly secure systems (B2, B3, and A1), because so many systems evaluated at lower levels of

security are structured to try to meet this requirement. Trusted Facility Management Requirements are given in Table 7.6.

Table 7.6 Trusted Facility Management Requirements

Class	Requirement
B2	The system must support separate operator and administrator roles.
B3, A1	The system must clearly identify the functions of a security administrator, whose job role is limited, as much as possible, to performing security-related functions. The security administrator and the system administrator/operator need not be different people necessarily, but their roles must be clearly divided. Whenever the system administrator assumes the role of security administrator, this role change must be audited. The security administrator's job is to perform security functions; non-security functions are strictly limited.

7.7.4.5 Trusted Recovery

Trusted recovery ensures that security is not breached when a system crash or some other system failure occurs. Trusted recovery is required only for B3 and A1 systems.

7.7.4.6 Security Testing

The security testing assurance requirement is closely related to the test documentation requirement (see "Test Documentation"). The system developer tests all security features, ensures that the system works as described in the documentation, and documents the results of testing these features. The NCSC evaluation team then undertakes its own testing. Security Testing Requirements are given in Table 7.7.

Table 7.7 Security Testing Requirements

Class	Requirement
C1	Tests must show that the security mechanisms work as described in the documentation, and that there are no obvious ways for an unauthorized user to bypass or defeat these mechanism.
C2	Additional tests must search for obvious flaws that would allow violation of resource isolation or permit unauthorized access to audit or authentication data..
B1	Additional tests (performed by an expert team) must search thoroughly for system flaws that would allow an unauthorized user to defeat discretionary or mandatory access controls or that would cause the TCB to enter a state in which it's unable to respond to user requests. (This is a denial of service problem). All such flaws must be corrected or neutralized (i.e., the system must be able to keep an intruder from doing damage if he does, in fact, break in as a consequence of one of these flaws).
B2	All such flaws must be corrected (neutralizing them isn't sufficient), and the system will then be retested. Additional tests must demonstrate that the TCB is consistent with the descriptive top-level specification (DTLS), described in the next section. At this level, the Orange Book states that the TCB must be found to be "relatively resistant to penetration".
B3	Testing must uncover no more than a few correctable implementation flaws (the exact number and type has to be worked out with the evaluation team), and there must be reasonable confidence that few additional flaws remain. At this level, the Orange Book states that the TCB must be found "resistant to penetration".
A1	Additional tests must demonstrate that the TCB is consistent with the formal top-level specification (FTLS), described in the next section

7.7.4.7 Design Specification and Verification

Design specification and verification requires a mathematical and automated proof that the design description for a system is consistent with the system's security policy. Design Specification and Verification Requirements are given in Table 7.8.

7.7.4.8 Configuration Management

Configuration management protects a trusted system while it's being designed, developed, and maintained. Configuration Management Requirements are given in Table 7.9.

Table 7.8 Design Specification and Verification Requirements

Class	Requirement
B1	The design documentation (see the section "Design Documentation") must contain either an informal or a formal model of the security policy (including subjects, objects, modes of access, security properties, and transitions from an initial system state to a secure system state).
B2	A formal security policy model is required. The design documentation must contain a mapping of all security properties to the security policy. At this level, an accurate (confirmed by testing) descriptive top-level specification (DTLS) of the TCB is also required. The DTLS is written in informal (i.e., not mathematical) language.
B3	The design documentation must show a one-to-one, unambiguous mapping between the DTLS and the TCB. The mapping must also show that the DTLS is consistent with the formal security policy model.
A1	A formal top-level specification (FTLS) is also required. The FTLS is written in a mathematical form that is precise and unambiguous. It is typically generated and processed via a formal specification and verification tool that's been endorsed as a testing tool by NCSC, as described later in this section. The design documentation must show a one-to-one, unambiguous mapping between the FTLS and the TCB and must also show that the FTLS is consistent with the formal security policy model.

Table 7.9 Configuration Management Requirements

Class	Requirement
B2, B3	Configuration management procedures must be enforced during development and maintenance. Configuration management must be enforced for changes to the descriptive top-level specification (DTLS), other design data, implementation documentation, source code, the running version of the object code, and testing resources and documentation. Tools must be available to generate a new TCB from the source code and to compare a newly generated version with the previous version.
A1	Configuration management procedures must be enforced during the entire system life cycle (i.e., during design, development and maintenance).. Configuration management must be enforced for all of the items listed for B2 and B3 systems, as well as security-relevant hardware, firmware, and software that modifies the formal model and the formal top-level specification (FTLS). Safeguards (technical, physical, and procedural) must protect the master copy of the TCB from modification.

7.7.4.9 Trusted Distribution

Trusted distribution protects a trusted system while the system is being shipped to a customer site. The trusted distribution requirement applies only to systems evaluated at the A1 level. The trusted distribution requirement has two goals: protection and site validation.

Protection means that at the vendor end (and during transmission from vendor to customer), trusted distribution ensures that the system that arrives at a customer site is the exact, evaluated system shipped by the vendor.

Site validation means that at the customer end, trusted distribution detects counterfeit systems (those sent by anyone except the legitimate vendor) or modified systems.

7.7.5 Documentation Requirements

The fourth and final set of requirements defined in the Orange Book is the set of *documentation requirements*. Documentation requirements are:

- Security Features User's Guide (SFUG).
- Trusted Facility Manual (TFM).
- Test documentation.
- Design documentation.

7.7.5.1 Security Features User's Guide (SFUG)

The Security Features User's Guide (SFUG) is aimed at ordinary, unprivileged system users. Typical topics include:

- Logging into the trusted system.
- Protection files and other information.
- Importing and exporting files.
- Dealing with system restrictions.

7.7.5.2 The Trusted Facility Manual

The Trusted Facility Manual (TFM) is aimed at system administrators and/or security administrators. It tells them everything they need to know about setting up the system so it will be secure, enforcing system security, interacting with user requests, and making the system work to its best advantage. The Trusted Facility Manual requirements are given in Table 7.10.

Table 7.10 Trusted Facility Manual (TFM) Requirements

Class	Requirement
All classes	Why security? Introduction to security concepts, rationale for using the security features described in the TFM (features that may seem onerous).
All classes	How do I administer identification and authentication features? How do I add users, set up authentication profiles, change password and enforce their use and protection? How do I help users with login problems?
All classes	How do I administer discretionary access controls? How do I protect system files using mechanism such as Access Control Lists, and what problems might users have using them?
C2 and above	How do I administer auditing capabilities? How do I set up my system for auditing, select the appropriate events to record, do selective auditing, maintain auditing files and media, and examine audit data?
B1 and above	How do I administer mandatory access controls? How do I set up the appropriate classifications and categories for my site, assign sensitivity labels, change user security levels, and deal with problems users may have accessing files and other system objects?
B2 and above	How do I generate a new TCB from source?
B3 and above	How do I start the system in a secure manner? How do I resume secure system operation after a system failure? How do I perform separate system administrator and operator functions?

7.7.5.3 Test Documentation

Good test documentation is usually simple but voluminous. It's not uncommon for the test documentation for even a C1 or C2 system to consist of many volumes of test descriptions and results. Test Documentation Requirements are given in Table 7.11.

Table 7.11 Test Documentation Requirements

Class	Requirement
C2 and above	Test documentation must describe the test plan, test procedures, and test results.
B2	Test documentation must include the results of testing the effectiveness of methods used to reduce the incidence of covert channels.
A1	Test documentation must include the results of mapping between the formal top-level specification (FTLS) and the TCB source code.

7.7.5.4 Design Documentation

Design documentation is a formidable requirement for most system developers. The idea of design documentation is to document the internals of the system's (or at least the TCB's) hardware, software, and firmware. Design Documentation Requirements are given in Table 7.12.

Table 7.12 Design Documentation Requirements

Class	Requirement
B1	Allows either an informal or a formal description of the security policy model.
B2	Requires a formal description of the security policy model and proof that it is sufficient to enforce the security policy. It also requires a descriptive top-level specification (DTLS) and a description of how the TCB implements the reference monitor concept.
B3	Requires that informal techniques show the TCB to be inconsistent with the DTLS and requires a mapping between TCB and DTLS elements.
A1	Requires a description of components internal to the TCB, and requires that a combination of formal and informal techniques show the TCB to be consistent with the formal top-level specification (FTLS) and requires a mapping between TCB and FTLS elements.

7.7.6 Problems with TCSEC

Although the Department of Defense Trusted Computer System Evaluation Criteria (TCSEC) [7.7], the "Orange Book" has existed for almost a decade, progress toward implementing systems based on it has been disappointing [7.4]. There are environments where operational concepts are ill-behaved with the concepts underlying the TCSEC. In most cases, the disconnects relate to the mandatory access controls. Mandatory controls work best in situations where data has easily determined, static security labels (classifications). There are situations where data dynamically changes security labels and where the only classified data results from the association of several pieces of information that are each individually unclassified. Examples of dynamically changing labels include scenarios where a planned event is classified prior to its occurrence but unclassified after it occurs.

Evaluated product features that allow "write ups" and support incorporation of trusted processes would reduce development risks. Understanding the types and frequency of classification changes and the amount of data involved are prerequisites for making sound system design decisions. Before pursuing approaches needing B level evaluated products, one should explore other alternatives such as restructuring the environment to be better behaved or pursuing system high rather than multilevel mode approaches.

7.8 MULTI LEVEL SECURE (MLS) SYSTEM DESIGN ISSUES

7..8.1 Design Issues

Attempts to design an operational system that effectively uses the security features can be extremely challenging. Difficulties often encountered include the lack of available higher level, secure components (e.g., networks, databases, windowing software, electronic mail), object granularity, performance and auditing [7.4].

Development activities are under strong encouragement to develop systems using commercial off-the-shelf software (COTS). Unfortunately, much of this software is not compatible with evaluated products, particularly with mandatory access controls. For example, office automation and electronic mail systems may not work unless all users and information are at the same security level. Attempts to operate with differing security levels either cause errors or frustrate users because of the need to change security levels. For example, such systems often maintain a user specific directory or database which must be stored in a security object of the evaluated product. The component will not work if the user attempts to use the component while operating at a security level other than the level of the directory or database.

The common belief is that the designers of secure applications could map objects at the application level onto objects of the underlying evaluated product. One difficulty is that the size of the objects can be grossly mismatched. Implementing small application objects using evaluated product objects intended for large collections of data may impact performance or force developers to augment the evaluated product with trusted software that essentially implements a new type of security object.

The issues of granularity and performance are interrelated. As stated above granularity can cause performance problems. Another performance issue that has occurred with several systems relates to logons in time critical systems. For some systems, time for one user to logout and another to logon at the same workstation can take several minutes. Most of the time is for application to load and initialize itself prior to being ready to accept input from the user. Often, the new user wants to resume exactly where the old user stopped. The process of destroying the old user's ongoing activity and rebuilding the same context for the new user seems unnecessary. A capability to change user accountability without disrupting basic mission processing is what the operational community desires. Unfortunately, none of the current evaluated products provide a rapid change of user accountability.

Another aspect of object granularity effecting operation comes up with auditing. Auditing needs to be addressed carefully. If the security objects at the applications level are not uniquely distinguished at the level of the evaluated product's objects, audit trail entries of the evaluated product may be useless.

Implementation of the security controls in applications software rather than risky attempts to use an evaluated product may do more to advance the eventual implementation of secure systems. While this may be a controversial

recommendation, the development of systems with proper functionality but without full assurance may do more to advance secure systems than continuing to wait for the "big bang" when a full set of secure components for building secure systems is available. Users and developers will gain knowledge and experience regarding the operation of secure systems. [7.4]

7.8.2 Secure System Composition

Secure commercial and military system components are increasingly available [7.5]. Worked examples of multilevel secure operating systems, database management systems, and local and wide area networks can be found in the open literature. The NCSC's evaluated Product List contains approximately 10 entries that have received B or A level ratings, and approximately 10 additional products at the B level or above are currently undergoing design analysis or formal evaluation. Today, the critical missing technology is the ability to create a composite system, using as components a heterogeneous collection of existing products, only some of which may be secure themselves [7.5].

In the following section, aspects related to distributed system security and their realization will briefly be mentioned.

Approaches

The approaches differ with respect to how system components are inter-related and composed, what properties may be expressed and proven for the resulting system, and whether the overall viewpoint of the specification is external (i.e., descriptive of the interface) or internal (i.e., descriptive of internal states).

The functions performed by a TCB include not only access control decisions, but also maintenance of the data related to access control (e.g., user ID's and clearances, access control lists, object labels, etc.), user identification and authentication, and auditing of security-relevant events. In a distributed TCB, such functions and data may be replicated across other components of the system, or partitioned among the components. It would appear that the specification of component roles in distributed systems involves two requirements: the ability to specify protocols for maintaining consistency and coordination among components, and the ability to specify partitioned data.

Shockley [7.8] provides a definition of domain (component) "independence", and observes that two components may be either independent, mutually dependent, or unilaterally dependent with respect to some set of correctness criteria: Given two domains, dA and dB, specifications of their interfaces, sA and sB, and demonstrations of implementation correctness, vA and vB, Shockley asserts that "Domain dA "depends (for its correctness)" on domain dB if and only if the arguments within vA assume (in whole or part) the correctness of the implementation of dB with respect to sB as a premise." Note that when more than two components are involved, unilateral dependence may be either circular or strictly hierarchical (partial ordering).

Communications protocols play an important role in distributed systems, not only for user communication, but also for communications among the distributed portions of the NTCB. The typical protocol "stack" represents a strictly hierarchical component structure, where each protocol layer is a component. In this case, dependence is similar to Lam and Shankar's (see [7.10, 7.11]) concept of linear hierarchies of modules in which a module "uses" the interface of a lower module, and "offers" an interactive to a higher module. (Note also, however, that the peer entities in a protocol stack may be mutually dependent). In addition to protocol specification, an approach such as Lam and Shankar's may be very suitable for situations such as those addressed in the TDI (where the concern is for incremental evaluation or "evaluation by parts"), and for extensible architectures such as described by Schaefer and Schell [7.12]. Hoare's CSP [7.13] permits both sequential composition (unilateral dependence) and parallel composition (mutual dependence and independence), as does the Abadi-Lamport Composition Rule [7.14].

It is frequently the case within distributed systems that the granularity of both subjects and objects is widely variable across components, and further, that the subject or objects of one component must be maintained or controlled in some specific relationship (including a relationship of labels) to the subjects or objects of other components.

Many of the specification approaches are based on an external view of systems and components, in which only the interface events and their associated "ports" are visible. Of the remaining approaches, which are typically state-machine descriptions, the MMS (Military Message System) model [7.15] provides a form of object granularity by means of the concept of containers, and Mclean's approach [7.16, 7.17] provides for one form of subsetting of subjects in the definition of a subsystem.

In composite systems, the amount of information encoded in security labels, and the particular form of the internal representation of those labels, will quite likely vary from component to component. The TCSEC requires a minimum of 16 hierarchical levels, and 64 non-hierarchical categories, although some systems have implemented considerably more than that. Even in situations where the number of levels and categories is the same for two components, the meanings assigned to the various levels and categories may differ. Reconciling such label inconsistencies during the integration of a composite system composed of pre-existing components is a critical and sometimes very difficult task which typically requires the creation of a label translation/mapping function.

Another labeling issue arises in systems where the granularity of subjects or objects varies, as described in the previous section. It may be desirable or even necessary to enforce a particular relationship among the labels associated with a set of subjects or objects.

In a composite system, it is quite likely that two or more components will each have a stated security policy that controls access of subjects to objects. In such systems, both issues of policy conflict, and issues of policy composibility must be addressed.

Policy conflicts arise between components if one component enforces a property which negates or weakens the policy of another component. One example would be a system in which component A enforces the Bell-La Padula property, which prohibits write-downs but not write-ups, whereas component B enforces a policy which prohibits both write-downs and write-ups. Depending on the particular system, this may be viewed either as a legitimate difference, or a serious policy conflict. Another example would be a system in which one component permits owner-users to modify the access permission matrix (the ACLs), but another component allows only the Security Officer to do so.

A concise specification of each component's security policy permits straightforward identification of conflicts such as these. However, analysis and resolution of the conflicts is not a technical issue, but rather a policy issue. In some instances, such policy conflicts may not be security weaknesses but rather a legitimate dual policy situation, while in other instances, the conflict may indeed be weaknesses, and some modifications to the policy and/or its underlying mechanism may be necessary as part of the system integration effort. In either case, once such analyses have been performed, one must then address the issue of policy composibility.

McCullough [7.18] discusses composibility for systems in which each component enforces the same security policy (the replicated policy type of system). Even here, composibility is not guaranteed: non-interference is not composable for non-deterministic systems, but restrictiveness is. McLean [7.16, 7.17] provides a framework for security policy models in which each model is distinguished by permissiveness for changing security labels (tranquillity "violations") which defines a form of sibling policies. It may be possible to extend this concept of approach [7.10, 7.11] and the Abadi-Lamport approach [7.14] permit the composition of components with arbitrary policies, which would be of benefit particularly for the third type of system (single distributed policy).

Within any complex, modernized system (distributed or not), it is frequently the case that those components which enforce the security policy depend on other portions of the TCB to supply secondary or supporting policies. By this we mean not only such functions as auditing, but a variety of functions and properties which become security-relevant by virtue of the fact that correct enforcement of the security policy depends on the function or property. This situation is particularly in evidence in distributed trusted systems. The distribution of portions of the system among two or more components of the system results in a need for communication among the components. Such communication usually requires mechanism to establish a TCB-to-TCB trusted path, and protocols which provide consistency of distributed security-relevant data, and concurrency control of security-relevant actions. Further, in those situations in which portions of the TCB are organized hierarchically (i.e., as a layered TCB), the policies enforced by higher levels of the hierarchy frequently depend on the "correct functioning" of lower levels. In a distributed system, actions which are traditionally modeled as atomic (uninterruptable) state transitions may actually require a sequence of such transitions, particularly where communication across a network is concerned. This constitutes an apparent shift in the "granularity of execution" when comparing the system viewpoint with the component viewpoint.

For the specification approaches, there are essentially two different ways of dealing with granularity of execution: either by showing the details explicitly, or by hiding the details via abstraction. For methods such as Hailpern-Owicki [7.19], the protocol involved is formally specified and verified. In approaches such as Hoare's communicating sequential processes [7.13], Lamport's logic of actions [7.23], Lam and Shankar's theory of modules and interfaces [7.10, 7.11], and the Abadi-Lamport composition approach [7.14], provision is made for hiding internal states of a component. These two approaches are complementary, rather than mutually exclusive, and both may be useful for any given system.

7.8.3 Security in a Distributed System

There has been much debate on the necessity of a fully MLS System. But the current trend in organizations processing voluminous data of versatile confidentiality put a more demanding requisite for MLS systems. There are some studies for a comprehensive Integrated Security System reported in Ref [7.20]. Given the availability of such technology we are posed with the question of how an unevaluated COTS product could be incorporated into an MLS environment built up with evaluated OS, GUI, CMW and DBMS products as shown in Figure 7.1.

In Figure 7.1, a personal computer and a Compartmented Mode Workstation (CMW) as clients, a host computer, a DBMS Server are interconnected through a LAN which altogether opens to the external world through a router. The corresponding security levels are also indicated for the components in this figure.

The labels corresponding to tangible (computers, physical storage devices and cabling) and intangible components (memory contents, operating system, user accounts and logical storage media) may be specified accordingly as in [7.21]. As it may be noted from the figure, clients may be CMWs or ordinary PCs. In the latter case special software is required to issue calls to the TCB.

In case, an unevaluated product such as a briefing tool is to be incorporated into the system, then each call for and from a Briefing Tool process window should pass through TCB to perform all security related operations. Such an environment has been described in [7.20] as the design of a Comprehensive Integrated Security System (CISS). Should an application attempt to open a file, CISS intercepts the request and determines whether the user has sufficient privileges for the operation as in Bell and La Padula model.

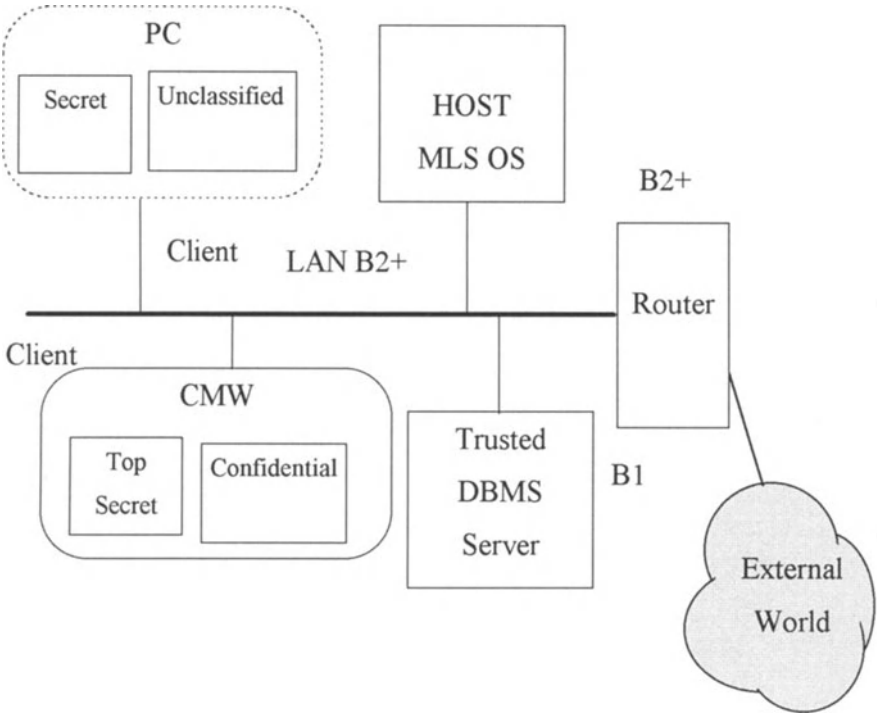


Figure 7.1 Incorporation of an Unevaluated Product in an MLS Environment

All files should have additional attributes specifying their security level as logical system objects. This is ensured by the MLS operating system. All file I/O request are monitored by the CISS.

The CISS API is implemented as a series of interrupt handler to monitor software interrupt activities. This may cause a performance degradation.

It should be noted that the CISS implementation should be evaluated to B2 or higher. Even when it is evaluated, the composition may have problems unless the criteria for composition are not taken into account.

Another example of an MLS system is described in [7.6] where the design of a complicated application, a multilevel graphical window system entirely in untrusted software is presented. [7.6] presents a design strategy for an MLS window system which can be implemented on a high-assurance trusted system without adverse impact on the assurance of the underlying TCB.

The possibility that the C^3I system is prone to attacks from external world and the fact that there may be subsystems operating at different security levels or complying to different security policies bring up the concept of firewalls and guard surveyed in the next section.

7.9 ALTERNATIVE SECURITY DESIGNS

Two concerns must be addressed in order to realistically discuss a balanced security design for our C^3I system:

- The need to exploit COTS software to the maximum extent posible
- The limited availability of MLS certified system

These concerns lead the practically oriented system designer to consider alternative approaches to the classic trusted system design approach as stipulated by the TCSEC [7] criteria. Two main initiatives will be presented below identifying these alternative approaches:

- Application Layer Security Functions
- Guard and Firewalls

7.9.1 Application Layer Security Functions

The Internet and its fantastic growth has made global information exchange a reality in a very short time. This wide area network infrastructure has one major characteristic that system designers are forced to face it is basically untrusted and insecure. Security can therefore not be an integral part of the network. It is inevitable that the networks which make up the global infrastructure will be considered to be insecure by their users and this is the essential driver of the need to protect the information on an end-to-end basis i. e. before it traverses the network. For these reasons, commercial pressures for effective means of end-to-end information protection will continue to grow and this will encourage improvements in the security of commercial products and hance a progressively better starting point on which to implement defence information systems like our C^3I system.

The recent key initiative in this area describes the security infrastructure to reside in the level between the network services and the (existing) application systems. An Application Independent Security Infrastructure (AISI) separates security components and modules from the applications which use those components. Its benefits are:

- Modifications to applications do not affect the security infrastructure
- Modifications to the security infrastructure do not affect the applications
- New applications can be developed which use the same infrastructure
- Greater trust can be achieved through modularization of security components
- All of these contribute towards lower cost and higher assurance systems

The elements of an AISI can be defined to include a Public Key Infrastructure (PKI), Cryptographic Application Programming Interfaces (CAP's), certificate management API's and the use and handling of labels and user attributes (capabilities, clearances and access controls)

The purpose of a Public Key Infrastructure (PKI) can be listed as:

- Generate public key certificates that bind the identity of the users and their public keys in a secure manner
- Provide users with easy access to the certificates of other users
- Provide users with easy access to circumstances (i. e. security policy) under which the certificates were issued
- Provide users with timely announcements of certificate revocations

As the requirement for application level security in the C^3I environment such as messaging grows, the need for a PKI is also going to grow. Although the specifics of particular security solutions and their algorithms will alter considerably over time, the underlying need for certificate distribution and checking is likely to continue for many years to come. Therefore a robust, flexible and re-useable PKI is considered an important component of a C^3I security infrastructure.

The area of openly available security API's is still in its infancy and are hindered by various export laws, licensing constrainnts and proprietary products. The goal is to deliver security services to applications with a minimum of changes to the application. In the longer term the use of security API's will be of benefit to all C^3I implementations, both in terms of reduced cost and increased assurance. Several factors limit the available of such solutions currently, to include immature technology, export legislation and few examples of true implementation.

Agreed international labelling standards for electronic information processing is another pre-requisite of an AISI. These must include label formats and semantics. A number of initiatives are being taken in this area but so far internationaly agreed standard has been developed.

7.9.2 Guards and Firewalls

Design of security aspects of a distributed system is treated like development of a software design specification. Threats are related by analogy with tests of software design and vulnerabilities are trated like design faults [7.21].

Organization and restructuring of functional areas into units with the same security levels and the need for information between units of different security levels have resulted in the development of firewall and guard concept [7.22].

Interoperability as described above requires MLS. With major wide-area networks (WANs) operating at the Sensitive but Unclssified (SBU), Secret, Top Secret, and Sensitive Compartmented Information (SCI) levels, this security partitioning causes operational deficiencies. It hampers interoperability, impedes data fusion, results in data that is less timely and accurate and results in duplication of facilities, with resultant increased operation and maintenance (O&M) costs.

A firewall is a computer, or a collection of computers, designed to protect an organization's internal computers from attacks. The firewall is tailored to the organization's needs, in that it implements the organization's security access policy. A guard is a specialized type of firewall that provides MLS capabilities.

Since guards are basically MLS firewalls, increased acceptance of firewalls has resulted in increased acceptance of guards.

Most guards today are typically based on class B1 or above platforms. Another main distinction is that guards are typically applied more to prevent the leakage of high-side information to the low side and to provide a trusted interface between environments operating at different security classification levels. Guards are being used to open doors that are too tightly shut, whereas firewalls are being used to close doors that are wide open. B1 or higher foundations protect the firewall by compartmentalizing the services provided by the firewall and thereby protecting the underlying operating system. Protection of network(s) behind the firewall is the responsibility of the firewall services, not the underlying firewall operating system. Firewall services such as application proxies are trusted to do their tasks correctly but are not given privileges that enable them to threaten the operating system.

There are different types of guards. Low-to-high guards support data flow between systems in which the sending system operates at a lower classification level than the receiving system. One shortcoming of low-to-high guards produced to date is that there has been almost no attention placed on protection against denial-of-service attacks from the low side.

Many man-in-the-middle guards are still in use, but are not covered in this survey because that solution has generally not proven to be operationally acceptable.

Automated release guards were initially developed for several intelligence systems and are seeing increased use within the intelligence community. The crucial improvement over the man-in-the-middle guards is that human reviewers remain at their normal workstations and that the review is performed as a natural part of their normal jobs.

Automated review guards automatically review the data and might also change the data to lower its classification. Operational versions of such software have tended to be application-specific.

7.9.2.1 Alternative Solutions to Guards

The main cause of interoperability is the islands of the system operating at different security levels. Often, the best way to solve this problem is to run the systems at the same security level which brings the possibility that the security level of a system to be changed.

There are two ways to change the security level at which system operate:

- Reduce the classification of certain data.
- Reduce the classification of systems by removing small percent of data that forces a high classification for the system as a whole.

A number of methods are in use that are variously referred to as air gaps and sneakernets. These are rekeying which is the manual rekeying into the receiving

system, media transfer (e.g., floppy disks), and periods processing in which all disks are removed and the system is scrubbed, so that the system but not its data, can be reused at a different security level.

Another alternative is MLS workstations. MLS workstations are different from guards in that they can provide interactive MLS access that guards cannot. Security guards concentrate on supporting data flow between systems, whereas MLS workstations provide interactive access between systems. A user at an MLS workstation essentially can be a full-capability user of two systems at two different security levels and can transfer data between the systems. One MLS workstation thus replaces two system high workstations, while providing a look-and-feel similar to that of non-MLS workstations.

MLS workstations also support a different concept of operations (CONOPS) than guards. An MLS workstation involves a user who simultaneously works in both environment. MLS workstations provide a downgrade capability that enables them to support a human review-based guard function.

7.9.2.2 Useability of Guards

A high assurance foundation, in itself, does little to protect the system(s) behind the guard. The guard's ability to protect internal systems is dependent on the quality and design of the specific guard services and the policies that they implement.

The MLS mode of operation still is needed for some MLS DBMSs, servers, and other capabilities, but it clearly is not the only way in which MLS capabilities can be used. That is, guards represent the use of multiple, independent checks as an alternative to relying upon one check.

Since there is little guidance on assurance for guards, service and agency planners should not develop new guards, but should use standard solutions that already exist. For example, several guards were designed to check for security errors before checking for communications errors. As a result, many communications errors were misinterpreted as being security errors.

Malicious software and covert channels have been at the root of many misguided risk avoidance efforts in the past. As a result, services allowing downloading software applications probably cannot yet be allowed through guards.

7.9.2.3 Integration and Certification for Guards

Engineering and integration (E&I) includes:

- Task definition and management
- Development of a CONOPS
- Software development and integration
- Hardware integration
- Security analysis
- Installation

- Documentation of the configuration
- Functional testing
- Initial operational support
- Evaluation of results.

Certification is short for certification and accreditation (C&A) which include:

- Development of a security CONOPS
- Coordination with accreditors
- Preparation of documentation supporting certification
- Performance of security testing and assessment.

Even when the underlying platform has been rated by the National Computer Security Center (NCSC), much certification is needed to assess the guard application, to assess modifications to the rated product, to assess the integrated system as a whole, and to assess compliance with additional requirements such as integrity and denial-of-service requirements and requirements unique to the local environment.

Guards need to be flexible to change with conditions. For example, it might be desirable during high alert periods to throw a switch and cut off a service or increase auditing. Typically, a guard accreditation encompasses both the guard and the underlying security rules that guards implement. Yet both change dynamically. Keeping it right is at least as important and has not received adequate attention.

It has become common for guard promoters to advertise advanced artificial intelligence (AI) techniques to analyze data and determine its classification. To date, such promises have proven woefully optimistic.

The underlying reality becomes more clear with an understanding of the difficulties of determining classification and releasability in DoD: Determining classification is a highly subjective, often dynamic judgment. This inherent ambiguity is a fundamental characteristic of many classified environments. Classified data simply does not automatically come in pre-painted security colors; comes in shades of gray. Because of difficulty of classifying output, data owners do not allow the system to support ad hoc queries. Without releasability annotated on the data, releasing officials often must contact the data originator to determine if release is permitted. This is operationally cumbersome, and often is impossible, given mission time constraints.

7.10 REFERENCES

[7.1] Russell D., Gangemi Sr. G. T., "Computer Security Basics", O'Reilly & Associates, Inc., USA, 1991.

[7.2] Varadharajan, V., "A Security Reference Model for a Distributed Object System and its Application", Proceedings of the 15th National Computer Security Conference, Baltimore MD, October 13-16, 1992.

[7.3] Sutterfield L., Schell T., White G., Doster K., Cuiskelly D., "A Model for the Measurement of Computer Security Posture", Proceedings of the 15th National Computer Security Conference, Baltimore MD, October 13-16, 1992.

[7.4] Price W. R., "Issues to Consider When Using Evaluated Products to Implement Secure Mission Systems", Proceedings of the 15th National Computer Security Conference, Baltimore MD, October 13-16, 1992.

[7.5] Hemenway J., Gambel D., "Issues in the Specification of Secure Composite Systems", Proceedings of the 15th National Computer Security Conference, Baltimore MD, October 13-16, 1992.

[7.6] Mayer F.L., Padilla S.J., "An Example Complex Application for High-Assurance System", Proceedings of the 15th National Computer Security Conference, Baltimore MD, October 13-16, 1992.

[7.7] Department of Defense, DoD Standard: DoD Trusted Computer System Evaluation Criteria in Specific Environments, DoD 5200.28-STD, December 1985.

[7.8] Shockley, W.R., Dockmaster TDI Comments forum entry #221, August 24, 1990.

[7.9] Amoroso E., Kleppinger W.E., Majette D., "An Engineering Approach to Secure System Analysis, Design, and Integration", AT&T Technical Journal, Vol. 73, No. 5, pp. 40-51, Sep./Oct. 1994.

[7.10] Lamport L., Shankar A.U., Woo T., "Applying a Theory of Modules and Interfaces to Security Verification.", Proceeding of the IEEE Symposium on Research in Security and Privacy, 1991.

[7.11] Lamport L., Shankar A.U., "A Composition Theorem for Layered Systems.", Proceeding of the 11th International IFIP WG6.1 Symposium on Protocol Specification, Testing, and Verification, Stockholm, June 1991.

[7.12] Schaefer M, Schell R.,"Toward an Understanding of Extensible Architectures for Evaluated Trusted Computer System Products.", Proceeding of the IEEE Symposium on Security and Privacy, 1984.

[7.13] Hoare C.A.R., "Communicating Sequential Processes.", Prentice/Hall International, London, 1985.

[7.14] Abadi M, Lamport L., "Composing Specifications.", Research Report 66: Digital Systems Research Center, October 1990.

[7.15] Landwehr C.E., Heitmeyer C.L., McLean J., "A Security Model for Military Message Systems.", ACM Transactions on Computer Systems, Vol. 2, No. 3, August 1984.

[7.16] McLean J., "The Algebra of Security.", Proceeding of the IEEE Symposium on Security and Privacy, 1988.

[7.17] McLean J., "The Specification and Modeling of Computer Security.", Computer, January 1990.

[7.18] McCullough D., "Noninterference and the Composibility of Security Properties.", Proceeding of the IEEE Symposium on Security and Privacy, 1988.

[7.19] Hailpern B., Owicki S., "Verifying Network Protocols Using Temporal Logic.", Proceeding of the IEEE Conference on Trends and Applications: 1980 Computer Network Protocols.

[7.20] Muftic S., Patel A., Sanders P., Colon R., Heijnsdijk J., Pulkkinen U., "Security Architecture for Open Distributed Systems", John Wiley & Sons Ltd. UK., 1993.

[7.21] Bhaskar K., "Computer Security: Threats and Countermeasures", NCC Blackwell Ltd, UK., 1993.

[7.22] Defence Information Systems Agency: Joint Interoperability and Engineering Organization, Security Guard Survey, 1994.

[7.23] Lamport L., "A Temporal Logic of Actions.", Research Report 57: Digital Systems Research Center, April 1990.

CHAPTER 8

SYSTEM MANAGEMENT

8.1. REQUIREMENTS AND OBJECTIVES

A fundamental assumption made in designing a C^3I system, may it be for civilian or military applications, is that it will meet all the requirements of its user(s) and that it will have a distributed architecture to correspond to the geographically distributed user organization. It is vitally important that the system is always ready to serve its users under all foreseeable conditions which would prevail in peace, tension, crises and war conditions.

We shall call the system that is used to ensure continuity of service, System Management or System Surveillance and Control Subsystems (SURCONS).

The function of SURCONs is to provide:

- Surveillance (monitoring),
- Control and maintenance,
- Service and resource provisioning,
- Support to planning activities and
- Support to system implementation and development

for all C^3I system elements with a view to maintaining maximum performance under changing operational and environmental conditions, natural and man-made stresses, disturbances and equipment distruptions.

Surveillance and Control considerations for any system, particularly of a military kind, operating in a common user multiple interest configuration must take into account not only the facilities making up the system, but also the users of the system and environment in which the system operates [8.1]. SURCONS will therefore be required to:

a) ensure the required system availability for its users during times of peace, tension, crisis and war,

b) maintain the system during the same conditions at a stipulated level of performance, i.e. grade of service, quality of service,

c) provide and restore service to users in their appropriate priorities under all operating conditions,

d) support centralized and decentralized modes of operation,

e) have a level of survivability at least equal to that of the subsystems which it is controlling,

f) have associated with it security measures designed to assure a maximum of protection against all anticipated active and passive adversary actions directed at deteriorating network performance or integrity, and

g) have an organization with adequate manning, sufficient control and test equipment's and other facilities to meet the above requirement.

The SURCONS must support in a time efficient manner the following control functions:

a) The status and performance monitoring of all system elements, in order to identify where and when faults and degradation's are occurring, or are likely to occur in the network, ideally before these affect overall system performance and service provided.

b) Maintenance of the required service and performance through timely fault detection, location, isolation, traffic and configuration management and repair as appropriate. In particular, the capability to restore service to designated critical users within minutes of a failure of a system element.

c) Maintenance of current data on the users and the status of resources and assets which may be used to support and restore service under stress condition.

d) Collection of selected data on traffic flow, equipment utilization and failure rates, user to user and link availability's, grade of service, and message delivery times which will be used for configuration and traffic management. This data is also to be used in support of longer term system planning and management functions.

e) Database management and the audit of hardware and software resources.

f) Effective planning and direction of logistics and maintenance activities on the system.

g) The electronic distribution of encryption keys if required.

h) Prevention of unauthorized access and interference to system processors, data and programs stored on the processors and their peripherals.

i) Co-ordination with other and interconnected systems control organizations at appropriate levels, and furnishing the agreed reports and status information to these systems.

System Management functions are performed at the three organization levels of (office (node), region, organization-wide) with different characteristics and scope:

- At the Node level, the system management mainly consists of: end-user support, system operations, network management, HW/SW maintenance and support,

- At the Regional level, the following System Management functions are required: regional data administration, regional configuration management, regional software acquisition, integration and customization, regional software maintenance, training,

- At the Organization level, System Management consist of: organization-wide data administration, acquisition, integration and customization of organization-wide off-the shelf applications, development and maintenance of applications with organization-wide applicability, organization-wide network management, organization-wide configuration management.

Standards and products for System Management exist only for the (Network) System Management and Management of Distributed Systems. The ISO, OSI System Management, ISO Common Management Information (CMIP, CMIS), CCITT X.700 standards are recommended for the long-term.

The most practical choice for the near-term is the SNMP protocol. It is part of the Internet protocol suite and provides a full robust basis for the management of network connected devices. The OSI common Management Information Protocol (CMIP) will require some years of research and development before mature products are available.

8.2. SURVEILLANCE AND CONTROL SYSTEM (SURCONS)

SURCONS includes System Operations, System Security Administration, Database Administration and System Support..

8.2.1 System Operations

System Operation comprises the following functions:

- **WAN Interface Management and Control.** - This function includes:
  - traffic monitoring
  - fault analysis
  - network initialization
  - configuration control
  - message handling control

- **LAN Management and Control.** - The LAN functions are equivalent to those required for the WAN.
- **Node System Control.**- This function comprises all tasks to keep the overall node system operational, including:

    - system administration services
    - operating services such as back-up/recovery, restart, error handling, system monitoring, resource administration
    - help desk functions.

These functions are supported by specific applications and services, executed by the System Management staff from dedicated workstations connected to the system nodes.

The architectural structure of SURCONS is depicted in Figure 8.1. This structure which is in conformity with the distributed architecture adopted for the C^3I system itself is controlled at three different levels. SURCONS is independent from the C^3I system but is connected to it at certain points and uses the same communication subsystem.

In Fig. 8.1, physical connections between the control centers and the system elements are shown in bold lines whereas functional connections are represented with broken lines.

There shall be equipments such as servers, user terminals (e.g. CMW) and peripheral equipment as well as Technical Control Centers to Control them which are all connected to the Local Area Networks (FDDI-2) which are to be established at every node. The built-in maintenance/operation and management functions (MF) in every system software and hardware element monitors its control functions and transfers the data obtained via the local area network to the Technical Control Centre (TCC). The Technical Control Centre processes the data it receives from the Management Functions (MF) and presents it in a suitable format to the operation and maintenance personnel in text, visual or audio form. It also transmits to the higher level important alarm and data related to the functioning of the system.

The Control System should support the control functions in two different nodes; "Centralized Control" and "Decentralized Control". Centralized Control improves system efficiency enhance restoral and reconstitution capability and reduces operation and maintenance cost. More specifically it reduces manning cost. Decentralization on the other hand improves survivability without which even limitless capacity and capability is meaningless in stress (e. g. wartime) conditions.

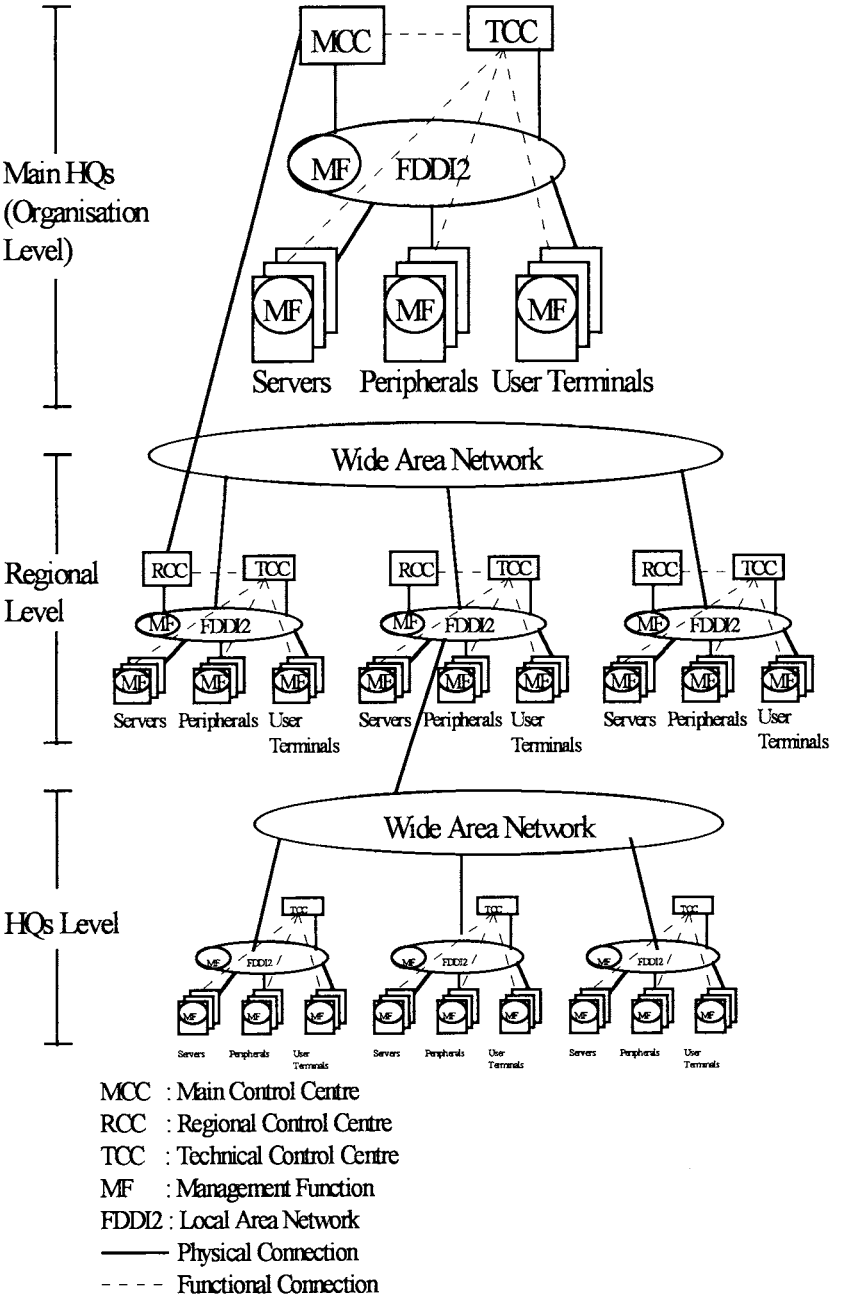


Figure 8.1 System Surveillance and Control Architecture

The Control System should support the control functions in two different nodes; "Centralized Control" and "Decentralized Control". Centralized Control improves system efficiency enhance restoral and reconstitution capability and reduces operation and maintenance cost. More specifically it reduces manning cost. Decentralization on the other hand improves survivability without which even limitless capacity and capability is meaningless in stress (e. g. wartime) conditions. The SURCONS concept is therefore based on a hierarchical structure supporting both centralized and decentralized control operations.

In normal times, the system wide quality of service on resource management activities will be relatively low and time delays in implementing these activities and activating control actions will not be critical. It is therefore possible to coordinate any operation throughout the SURCONS echelons with the centralised mode of operation. However, under stress conditions, service and resource management activities and control measures have to be implemented very quickly and sometimes without the time to wait for centralised co-ordination. Further as a consequence of system fragmentation control may also be fragmented and some of the SURCONS resources may not be available to support a centralised operation. Thus system operations have to be functioning autonomously at local and regional levels.

Decentralized control requires distributed control cells, distributed control functions and distribution of the databases to locations close to the relevant users and their operational needs. SURCONS will be designed so that the application functions and information are available (i. e. distributed) to all control levels.

In normal times, i.e. centralised mode, the Management Functions (MF) at the regional level (RCC), primarily provide surveillance and control functions for the assigned region to initiate or perform various operations. The MFs will also process the surveillance data to form an input in contribution to the overall system performance to the MCC. The MFs at the MCC level will collect the pre-processed surveillance data from regional control centres to analyse these data to establish an overall system performance view. The MCC MFs will also perform service and resource management functions. System configuration and system management decisions will be taken at the MCC level for the decisions involving several regions and at an RCC for the actions specific to a region.

Since the system is designed to function in normal as will as stress conditions including war-time conditions, special attention must be given to the principles of distribution of control responsibility for any given system element.

As a basis, the control authority must reside with the local TCC. If access is available to another authenticated TCC or an RCC, the TCC will allow its control to be transferred. The same applies to the RCC that will transfer its control upon request to the MCC or to another RCC that is correctly authenticated.

In the presence of severe stress, damage or suspected compromise, the lower level centres will automatically take back the control and will have to be equipped with all means locally to perform all of its required SURCONS functions.

The functional flow diagram of the monitoring and control system is given in Fig. 8.2 which shows the data, reports, service instructions and service demands related to the system control functions which are exchanged between the control centres at various levels. Reports relating to the data about the system conditions and instructions executed are sent from lower to higher level control centres. Upon receiviving the above mentioned data and reports, the higher level control center assess them and send to the lower levels appropriate service instructions and measures for execution.

For control purposes all the control centres should be equipped with high performance Compartmented Mode Workstations (CMW) having identical hardware and software. In addition the workstation incorporated in the local area network of the Main Control Centre should be loaded with SNMP software (Simple Network Management Protocol) for network management.

All information enchange related to system control should be encrypted and a separate workstation should be provided at every user location for security administration.

Standards for System Control and Management

All the workstations mentioned above should use system control software which has been and is being developed by ISO OSI and CCITT and should have all the necessary databases which they should keep continuously updated.

The system management and control products and standards that are available at present are only for network management and distributed system management. The standards that are recommended for the long term are ISO OSI Common Management Information Protocol (CMIP) and Common Management Information System (CMIS) and X.700 standards.

The system control and management related standards which are being developed are presented in Table 8.1. The most practical approach for the network environment is SNMP (Simple Network Management Protocol) which is a part of The Internet Protocol group.

SNMP has been designed to work with TCP/IP. The original SNMP product did not have security features and was replaced in 1992 with SNMP2 which has them.

SNMP measures the error performance of the network and sends the error data to the central control centre. OSI Common Management Information Protocol (CMIP) has not yet been fully established and is in need for further development.

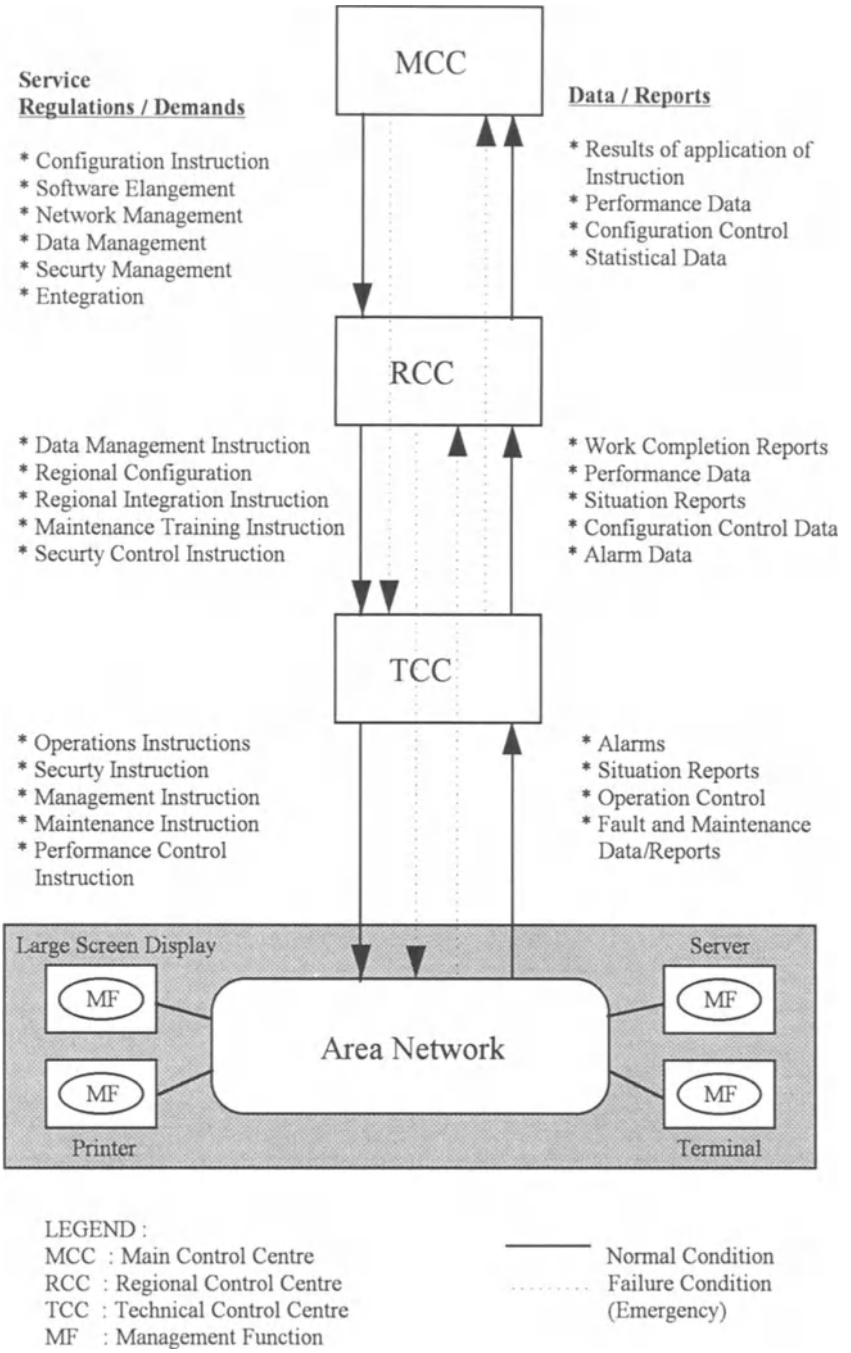


Figure 8.2 Functional Flow Diagram of the Monitoring and Control System

Table 8.1 System Control and Management Standards

Standards No.	Subject	Type	Cross-Reference (Related STANAG) *
ISO 10040	System Management General Analysis	ISO/CCITT	4250-4 4407
10165	Management Information Structure		
ISO 10165-1	Management Information Model	ISO/CCITT	4407
ISO 10165-2	Description of Management Support Objects (GDMO) CMIS/CMIP	ISO/CCITT	4407
DIS 10165-5	Generic Management Information	ISO/CCITT	
DIS 10165-6	Management Inf. Compatibility Statement Requirements & Guiding Aspects		
ISO 9595:90	General Management Knowledge Service	ISO/CCITT	4407
ISO 9596:90	PICS Performance	ISO/CCITT	4407
10164	System Management Functions	ISO/CCITT	
ISO 10164-1	Object Management Function	ISO/CCITT	4407
ISO 10164-2	State Management Function	ISO/CCITT	4407
ISO 10164-3	Objects & Features for Showing Relations	ISO/CCITT	4407
ISO 10164-4	Alarm Report Function	ISO/CCITT	4407
ISO 10164-5	Event Report Management Function	ISO/CCITT	4407
ISO 10164-6	Log Control Function	ISO/CCITT	4407
ISO 10164-7	Security Alarm Report Function	ISO/CCITT	4407
DIS 10164-8	Security Authorisation Observing Function	ISO/CCITT	4407
DIS 10164-9	Objects & Features for Access Control	ISO/CCITT	4407
DIS 10164-10	Account Keeping Measurement Function	ISO/CCITT	4407
DIS 10164-11	Work Load Monitoring Function	ISO/CCITT	4407
DIS 10164-12	Test Management Function	ISO/CCITT	4407
DIS 10164-13	Summary Function	ISO/CCITT	4407
ISO 10733	Elements of MI Related NL Standards	ISO/CCITT	4407
ISO 10737	Elements of MI Related TL Standards	ISO/CCITT	4407
ISO 10742	Elements of MI Related DL Standards	ISO/CCITT	4407

* STANAG : NATO Standardisation Agreement

8.2.2 System Maintenance

8.2.2.1 General

In a system as vital as any C^3I it is essential that service is maintained for 24 hours continuously and without interruption. This is made possible by an effective maintenance support and its continuous control. To achieve this aim it is necessary to monitor all system elements continuously in order to find out all fault conditions and take the necessary preventive measures before they cause system interruption.

This involves hardware and software maintenance which are described below.

8.2.2.2 Software Maintenance

a. Concept

- Software maintenance may be carried out either in-house using the establishment's own resources or by on-site contractors . Regardless of how it is done, it is important to have a "Software Maintenance Section" within the organisation who, in fact, decides how maintenance is done.
- Software maintenance covers software support, configuration control, quality control, and verification and testing.
- The commercial programs (COTS) used in the system are normally maintained by the companies who produced them
- Programs produced in-house are maintanned by the Software Maintenance Section

b. System Support

- Data base management is to be effected from one central point.
- The manpower and the automatic support tools necessary for operation and maintenance are to be provided at every node of the system.
- The system security administration is to be performed by personnel at different levels of the system. They will be equipped with security monitor and control facilities.
- There shall be "help desks" at every centre with integrated software, hardware and communication related support capability.

c. Software Tool Support

There is a need for the following automatic tools for supporting maintenance activities.

- Tools for program / project management,
- Configuration management tools,
- Software engineering / computer aided software engineering (CASE) tools,
- Tools for quality control

8.2.2.3 Hardware Maintenance

a. Concept

- Hardware maintenance is to be, performed at local and depot levels by contractors.
- There shall be "help desks" at every centre with fault finding capability.
- Faulty system elements which can not be repaired shall be replaced with spares under help desk control.

b. Inventory

Inventory of the system hardware is to be maintained at every level by the responsible units in accordance with established procedures.

c. Spare Parts

- Adequate amount of spare parts should be provided under the control of help desks.
- The level of spare part provision should be consistent with the requirements of availability and survivability of the system.
- Minimum level of spare part stocks to be kept should be determined taking into account the statistics of material useage.
- Critical parts and parts with the highest rate of failure should be kept locally and less critical parts should be maintained by the contractor in his local depot.

d. Replacement Concept

- The average lifetime of system hardware may be taken as 10 years.
- Replacement of parts should be effected according to a plan.
- One should pay attention to the drawing up of long-term guarantee agreements for parts which are more economically provided withim a guarantee framework or the maintenance of which is beyond the capability of the organisation concerned.

8.2.2.4 Resource and Service Provisioning

Resource and service provisioning aims at maintaining the service provided to the users at the highest possible level by effectively using the capabilities of the existing hardware, software communications and human as well as material resources. What is important here is to transform the service demand of the users to the resource provisioning steadily to meet the demand. To achieve this, it is necessary to maintain databases for resource and service provision activities as well as for user and resource related information.

8.2.2.5 Data Administration

Data Base Administration is the day to day process of ensuring that the standard data definition and structures developed as part of Data Administration are implemented and maintained in operational software. Data Base Administration

also ensures integrity of data in the system through backup andrecovery procedures, application of data ownership rules, maintenance of database access permissions and audit of database updates.

Data Base Administration requires its own work position on the system with special software tools as well as its own support tools in the software maintenance and development environment.
In order fulfill these requirements, the concept of an Information Resource Dictionary System (IRDS) is introduced. The structure of IRDS and recommendations for its use are given in Sec. 5.2.4.2 of Chapter 5.

8.2.3 System support

Software Development and Maintenance should be supported by a suitable support environment. A standard ADP environment should include the following software components.

- CASE tools
- operating system interface for application programming
- third generation language development environment
- database management system
- fourth generation environment
- user interface management system
- geographic information system
- text and graphics editing tools
- software configuration management tools

The tools have to be compatible (partly identical) with the software modules of the operational node-systems.

The software Development Environment should be implemented on a node system logically independent from the target system itself.

8.2.4 System Security Administration

Security Administration comprises all means to implement the identification, authentication and authorization of users and to provide security audit services.

- **Security Administration** - The function deals with the maintenance and control of security attributes in the system including identifiers, authentication information (password and labels), and access right

information. It sould be possible to perform these functions from a single workstation even in a real implementation. Security administration functions will be distributed regarding the different (possibly heterogeneous) components such as LAN, servers and workstation.

- **Security Audit** - This function includes the means for selectively recording user activities which are security relevant (audit trail) and to audit individual user actions selectively.

These functions are supported by specific applications and services, executed by the Security Management staff from dedicated workstations connected to the system nodes.

8.3 REFERENCES

[8.1] Ince, A. N., et al , "Planning and Architectural Design of ISDN", Kluwer Academic Publishers, Boston, 1995.

CHAPTER 9

SYSTEM COSTING AND IMPLEMENTATION

9.1 SCOPE

Having defined the architecture and configuration including the dimensioning of the C^3I system we now have to cost it in terms of capital and recurring costs and draw up an implementation plan over a time scale consistent with the user requirements and within funding and other constraints set at the beginning of the design process (see Sec. 1.5). If the cost calculated exceedes the financial limit set then the design process is iterated as shown in Fig. 1.3 of Chapter 1 by curtailing some of the requirements relating to, for istance, security, survivability and, the users and functional areas to be served. This iterative process must involve both the user/owner of the system as well as the designers so that the final requirements are stated in terms which would lead to a feasible, cost-effective system capable of meeting the operational requirements including the various constraints as shown in Fig. 1.3.

In this chapter we shall describe a cost methodology for costing the overall C^3I system including estimation of the application software to be developed and of the cost of operation and maintenance of the system.

We shall also outline an implementation strategy based on commonly used and proven technology using the standards recommended in the previous chapters of the book and the principle of "evolutionary development and acquisition".

It is, of course, appreciated that the total capital and recurring costs as well as the implementation phasing and time-table would depend, among other things, on the precise user requirements from which the size, capacity and capabilities of the system would be determined as explained in the previous chapters. In order that the reader may gain an insight into the order of costs involved in implementing a C^3I system over a time period which is in accord with the user needs and priorities we shall postulate a network which shall be, as before, of a strategic military kind. Since cost is determined by the number of subsystems and their constituent elements (each with a unit price) which make up the total system, the reader can cost and plan the implementation of his own system using the costing and implementation methodologies described and the unit prices provided for both hardware and software products outlined in this chapter.

We shall assume that our C^3I system would be as outlined in Fig. 3.2 consisting of some primary nodes (those in the main and alternate HQs and in the force commands) and smaller secondary nodes (those in the subordinate commands) interconnected by MAN and WAN networks. The civilian equivalent of this system would be as shown in Fig. 1.2 of Chapter 1. In the following sections we shall cost a primary node (that in the main HQ) which is depicted in Fig. 9.4 including all the hardware products used as well as the software products which can be bought off the shelf and those to be developed. Software products being scalable and portable would be used for all the other nodes taking into account the software modification costs and lisence fees to be paid for the COTS products in each node. The reader can then cost any node of any size using the data provided in this chapter. In addition to the investment costs we shall also provide costs and expenditures associated with the surveillance and control organization including O&M costs.

9.2 COST METHODOLOGY

There are two main categories of cost/expenditure that make up the total cost of a C^3I system; namely "investment or capital cost" and "operations and maintenance cost or recurring cost" as shown in Fig. 9.1. Total investment costs (non-recurring costs) will include expenditures for hardware, commercial software, software development and other additional expenditures. Recurring expenses will include expenditures for hardware operations and maintenance, software maintenance, SURCONS organization and Project Office expenses and Public Network line charges.

Hardware and commercial software cost figures used in the sections below have been estimated using:

i) Product prices from the manufacturers catalogs and local company resperestatives,
ii) Prices from NATO documents,
iii) Prices obtained from procurements in which the authors have been involved.

The commercial software costs are estimated on the basis of the maximum number of users that will use the product. License fees for the Data Base Management System (DBMS) should take into account all the users connected to all the nodes of the C^3I System.

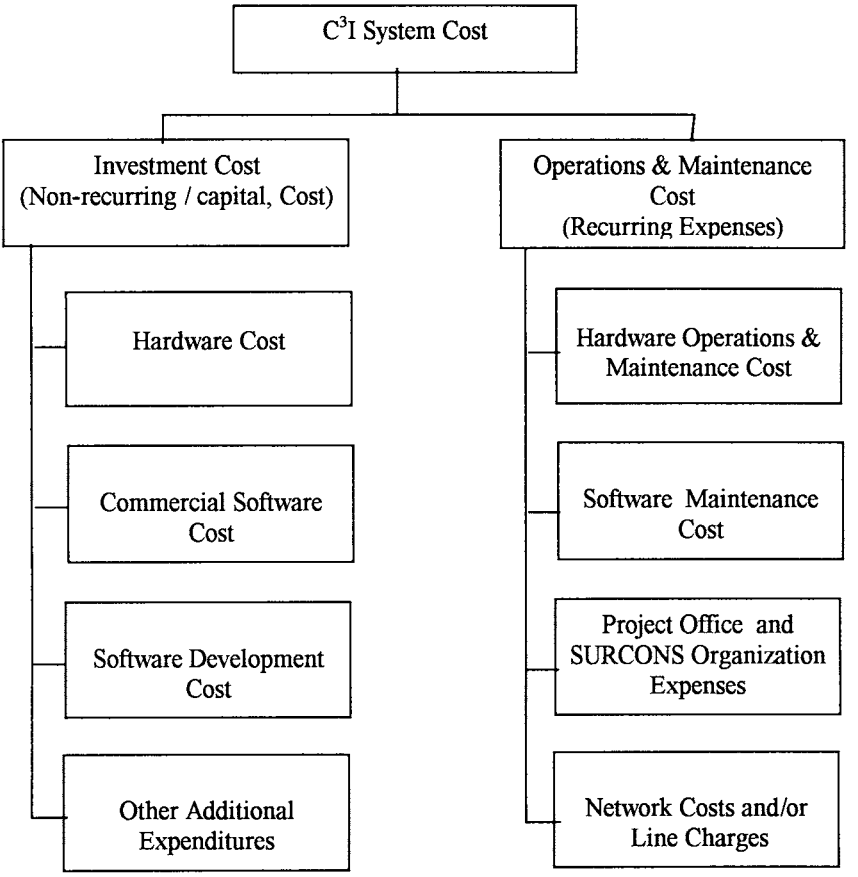


Figure 9.1 C^3I System Cost Methodology

9.2.1 Investment Costs

Non-recurring capital costs are considered in three categories:

- Hardware and Commercial Software (COTS products) Cost,
- Development Cost of the Application Software,
- Miscelleneous Additional Expenses.

9.2.1.1 Hardware and Commercial Software Investment Costs

The types, quantities and approximate unit prices of hardware and COTS software products required for a main headquarters node (primary node) where a staff of about 200 persons are estimated to work, are given in Tables 9.1 and 9.2.

The quantities of the required hardware and software products are estimated from the number of staff working per shift and the volume of work they generate using the data in Tables 2.2, 2.3 and 2.4 of Chapter 2.

The total hardware and COTS software investment cost estimates can be obtained by calculating the cost of each C^3I node of the system based on the information given in Tables 9.1 and 9.2.

Table 9.1 Hardware Investment Costs For a Main Headquarters Node (Example)

Hardware Units	# of Units	Remarks	Unit Price (US $)	Total Cost (US $)
DBMS Server	1	• Supports 60 Terminals • Symetric Multiprocessor • 256 MB Main Memory	500.000	500.000
GIS Server	1	• Supports 20 Terminals • Symetric Multiprocessor • 256 MB Main Memory	200.000	200.000
CAX Server	1	• Supports Exercise Activities • Symetric Multiprocessor • 256 MB Main Memory	500.000	500.000
Decision Support Server	1	• For Decision Support Activities • Symetric Multiprocessor • 256 MB Main Memory	200.000	200.000
Briefing and Document Server	1	• Supports 100 Terminals • Symetric Multiprocessor • 256 MB Main Memory	150.000	150.000
Applications Server	6	• Supports All Users And Applications • 256 MB Main Memory	200.000	1.200.000
Communications Server	1	• Supports All Terminals • 256 MB Main Memory	30.000	30.000
Message Processing Server	1	• Supports All Terminals • 256 MB Main Memory	40.000	40.000
A-Type Terminal (CMW) (For GIS and briefing preperation intensive users)	40	• 20 For Joint Operations Center • 20 For Forces Operations Centers	10.000	400.000
B-Type Terminal (CMW) (For data base, text processing and display intensive users)	20	• 5 For Joint Operations Center • 15 For Forces Operations Centers	8.000	160.000

(Continued)

(Table 9.1 Continued)

Briefing Room Display Terminal	6	• 3 For Joint Operations Center • 3 For Force Commands Operations Center	20.000	120.000
Control Monitoring Terminal (CMW)	2	• One is spare	20.000	40.000
RAID Unit	1	• For DBMS Server	100.000	100.000
DAT Tape Unit	5	• One Each For DBMS, GIS, CAX, Briefing And Message Processing Servers	2.000	10.000
CD-ROM Unit	4	• One Each For DBMS, GIS, CAX and Briefing Servers	1.000	4.000
3.5' Floopy-Disk Driver	3	• One Each For DBMS, GIS, CAX Servers	100	300
Optic Disk Unit	3	• One Each For DBMS, GIS, CAX Servers	2.000	6.000
Line Printer	6	• One Each For Applications Servers	5.000	30.000
Laser Printer	6	• One Each For Applications Servers	7.000	42.000
FlatBed Color Scanner	1	• For Briefing And Document Server	2.000	2.000
A-Type Video Projection Equipment (5 m diagonal screen)	3	• For Joint Operetions Center	40.000	120.000
B-Type Video Projection Equipment (3 m diagonal screen)	3	• One Each For Force Commands Operations Center	35.000	105.000
Uninterrupted Power Supply (UPS)	1	• Feeds All Servers and Terminals • 350 KVA	350.000	350.000
Generator	1	• Emergency Power Supply • 350 KVA	350.000	350.000
B1 Secure LAN	1	• FDDI-II Dual Ring	200.000	200.000
DAC with 4 ports	15	• For Server Connections to LAN	1.000	15.000
DAC with 16 ports	10	• For Terminal Connections to LAN	3.000	30.000
Router	1	• For Connection to Communication Network	100.000	100.000
			GRAND TOTAL	5.004.300

Table 9.2 COTS Software Investment Costs
For a Main War Headquarters Node (Example)

COTS Software	Number of Licence	Remarks	Unit Licence Price (US $)	Total Cost (US $)
Operating System for Terminals	60	• For 60 Terminals	3.000	180.000
Operating System for Servers	15	• For 15 Servers	5.000	75.000
DBMS Software	100	• For 100 Users	2.000	200.000
GIS Software	20	• For 20 Users with Intensive GIS Activities	7.500	150.000
Briefing and Document Software	100	• For 100 Users	6.000	600.000
Decision Support Software Tool	20	• For 20 Users with Intensive Decision Support Activities	10.000	200.000
DB Applications Development Tools	100	• For 100 Users	1.500	150.000
GUIB Software	10	• For 10 Users to Develop and Test MMI	17.000	170.000
Message Processing Software	100	• For 100 Users	2.000	200.000
			GRAND TOTAL	1.925.000

9.2.1.2 Software Development Costs

Application software to be developed for the C^3I system, as explained in previous chapters, can be divided into projects and the number and productivity of the software engineers to be employed on the projects are determined considering the type and size of the software to be developed. The cost of each project can then be calculated by taking into account the number of personnel to be employed, and the duration of development.

Different productivity and field of expertise are assumed for different types of applications software to be developed such as Database Applications, Computation-oriented Applications and Graphics-oriented Applications.

Software Development Criteria

The complicated nature of,and interactions of events and individuals involved in Command and Control Systems,coupled with different software requirements for different functional areas (e.g. personnel, operations, logistic, etc.) and the fact that the required products are unavailable as off-the-shelf products make it essential that the following points are taken into account during software development in implementing a C^3I system:

- The software to be developed must be defined very clearly and precisely.
- The decision for the development of applications software should be based on the results of testbed/prototype studies and the final decision on the way ahead should be made together with the user.
- Special application software for functional areas should be developed domestically.
- Commonly used software should be associatively developed or provided centrally.
- Wherever possible, hardware independent COTS Software should be preferred and special software be avoided to minimize software development and maintenance costs.
- The software development methodologies should have features providing for re-usability of software (modular and structured software development).
- The software should be able to run in a variety of platforms ranging from personal computers to mainframes (use of standard compiler and standard programming language features).
- The technical personnel of the Organization should be fully involved in the development and be trained to be able to assure the responsibility of software maintenance in the subsequent stages.
- Follow-on software modifications and developments should be made domestically or by the organization staff.
- Software personnel should not be assigned to other duties.
- Software maintenance and personnel training costs should be minimum and the development process should be compliant with other application software development and designated criteria.
- Software development and maintenance should be made in a physically separate system from the operational system.

Engineering Techniques for Software Sizing

In order to be able to estimate the cost and time of the software to be developed the following method is recommended.

For determining the cost of software development, software engineering techniques are used [9.1]. A Software engineering technique that we recommend to use here is called "Quantitative Software Management" (QSM) which was established in 1978 by Lawrence Putnam who developed the core techniques during his career in the US Army. His methods of analysing project performance data and of estimating new work have now been used by over 200 organisations world-wide.

QSM's analysis and methods are described in [9.2]. QSM has a database of information relating to thousands of software development projects. This data is used to develop and tune the mathematical model which is the basis of the technique.

QSM uses simple high level measures to quantify projects. Any project which builds or modifies software and delivers a release into an operational environment can be calibrated or estimated. Software size is measured in lines of code (loc) counting new and modified code.

Using a database of project data, staffing as approximated by a Rayleigh curve is related to three project drivers: software size, process productivity and time pressure. Software size is measured directly in loc. Process productivity is stated using an abstract scale from 1 (very low productivity) to 40 (very high). This is the Productivity Index, PI. Time pressure is stated with the Manpower Buildup Index (MBI) on a scale from -3 (small team, long duration) to + 10 (extremely large team, rushed development).

Measuring project effort and elapsed time (together with software size) for a completed project allows the PI and MBI to be calculated. Conversely, by specifying software size and a reasonable PI, project time and effort can be calculated for a new project and alternatives generated using different MBI values (different size teams). QSM's tool, SLIM, performs calculations.

If the input parameters were known exactly in advance, SLIM would estimate project time and effort exactly. However the inputs are themselves estimates with a probability distribution over a range from a minimum value to a maximum. SLIM uses Monte Carlo simulation to translate this uncertainty in inputs into uncertainty in outputs.

The estimating method requires, as an input, a measure of the size of the software to be built. This reflects the size of the functionality of the system in conjuction with the techniques of construction (e.g. high level languages require less software than low level languages to deliver a given functionality).

Software size cannot be predicted exactly in advance of construction. It is reasonable to work with range estimates which reflect the degree of uncertainty in the software size. As with any estimating process, confidence is improved when different, independent sources or methods are used to generate alternative estimates.

Software size is generally measured in Lines of Code (LOC), using standard rules to define what should be included and excluded in the count. This is sometimes referred to as Effective Source Lines of Code (ESLOC).

Database Software Sizing

For estimating the amount of software required to create and use the databases, the complexity of the data is taken as the basis with the volume of data held in the database. Database Transactions are grouped into two for estimating process.

Definition and maintenance: The number of transactions required to set up and perform basic maintenance on the database (calculated as 2* number of tables + 0.2* number of fields). We take 100 loc to be a typical size of such transactions, written in a high level database language.

Archiving and recovery: The number of transactions required for system maintenance in archiving and recovery (calculated as the number of definition and maintenance transactions/10). In practice, archiving and recovery will be implemented as an integrated function covering all the databases. Again, 100 loc is a typical size.

For each of these two categories, four different types of transaction are considered as simple, medium and complex transactions seperately.

An example for software development cost is given below:

The total LOC (Lines of Code) of the application software to be developed is taken typically for a military C3 system to be 900.000 of which 500.000 LOC is for Database Applications, 60.000 LOC is for Graphics Applications, and 340.000 LOC is for Computation-oriented Applications. With this input and using the QSM methodology and assuming a 9-years implementation time the required number of personnel for software development and the distribution over the years are shown in Table 9.4 and the cost of software development is shown in Table 9.5. In the cost calculations annual cost of an expert is taken to be 80.000 US Dollars including all overheads.

If it were required to shorten the software development time than that given in Table 9.4, SLIM can be used to estimate, with a given probability of assurance, the appropriate size of team that would be needed and its cost to develop the software within the time required with a certain probability of assurance. As would be expected in this case, the mainpower required together with the associated cost would increase.

Table 9.4. Annual Distribution of Required Software Development Personnel

Years / SoftwareTo be Developed	1	2	3	4	5	6	7	8	9
DB Application SW			35	35	35				
Computation-oriented SW	5	8	8	5	4	4	4	4	4
Graphics-oriented SW			2	2	2	2	2	2	
Total	5	8	45	42	41	6	6	6	4

Table 9.5 Software Development Cost

Years / Software Developed	1	2	3	4	5	6	7	8	9	App. Cost (Million US $)
DB Application SW			2.8	2.8	2.8					8.4
Computation-oriented SW	0.4	0.64	0.64	0.4	0.32	0.32	0.32	0.32	0.32	3.7
Graphics-oriented SW			0.16	0.16	0.16	0.16	0.16	0.16		1.0
Total	0.4	0.64	3.60	3.36	3.28	0.48	0.48	0.48	0.32	13

9.2.1.3 Additional Costs

Non-recurring additional expenditure for total Hardware and COTS Software given as percentages of total system cost is shown in Table 9.6.

Table 9.6 Non-recurring Additional Expenditure

Additional Cost Items	Percentage of System Cost (%)
Integration	20
Training	13
Initial Spares	12
Documentation	6
Coding	2
Contingency	10
Total Additional Cost	63

9.2.2 Operations and Maintenance Costs

Operations and Maintenance Costs can be considered in four groups:

i) Hardware Operations and Maintenance Costs,
ii) Software Maintenance Costs,
iii) Project Office and SURCONS Organization Expenses,
iv) Circuit Rental Charges.

Operations and maintenance costs are determined annually, by taking into account the following points;

i) Hardware operations and maintenance cost and software maintenance cost should be listed annually in a convenient format during the system establishment.

ii) Manpower cost of the organization responsible for operating the C^3I Surveillance and Control System should be determined.

iii) Project Office expenses for the technical studies needed for system development should be estimated.

iv) Communication circuit rental charges should be determined annually by considering C^3I system configuration, number of circuits and rental charges.

During the system establishment phase, operations and maintenance costs for each year are determined as a percentage of the total investment cost spent up to the current year. Once the system is completely set up, the operations and maintenance cost for the succeeding years can be assumed to be fixed.

9.2.2.1 Hardware Operations and Maintenance Costs

Hardware Operations and Maintenance Costs for a C^3I System includes spare parts, limited replenishment spares, consumable supplies, contractor's logistic support, transportation, training and depot level maintenance.

Annual Hardware Operations and Maintenance Costs, are given in Table 9.7 as percentages of the total investmend cost based on the statistics obtained by the NATO Countries for their C^3I projects.

Table 9.7 Hardware Operations and Maintenance Costs

Hardware Operations & Maintenance Cost	**Percent of the Investment Cost**
Spare Parts	1
Limited Replenishment Spares	1
Consumable Supplies	0.5
Contractor's Logistics Support	2
Transportation	0.5
Training	0.5
Depot Level Maintenance	0.5
Total	**6 %**

9.2.2.2 Software Maintenance Costs

Software Maintenance Costs which are the expenses incurred for the maintenance of all the softwares developed and bought commercially, are used to procure new versions of the COTS software (Operating System, Data Base Management System (DBMS), Geographical Information System, Briefing and Document Processing Software, Decision Support Tools, DBMS Application Development Devices, Graphical User Interface Builder (GUIB) software, Message Handling and Processing Softwares) and to develop software for changing user requirements and contractor technical support.

In addition, the expenses necessary to integrate the developed software with the COTS products in service and to procure additional software to meet new user requirements or to upgrade the system have to be included in the Software Maintenance Costs.

It is generally accepted that the annual maintenance cost for UNIX based environment is approximately 15 % and for DOS based environment it is approximately 35 % of the Software investment cost. In case of an MLS UNIX based platform, 15 % of the total software investment cost can be taken to be the annual software maintenance cost.

9.2.2.4 The Expenditures of SURCONS Organization

It has been stated in Chapter 8 that a distributed comprehensive system such as a C^3I system requires a system management or system surveillance and control system (SURCONS) to ensure continuity of service. This requires very qualified personnel with expertise in different areas to manage, operate and control the system. An example of this organization and its personnel complement is shown in Table 9.8.

Table 9.8 Man-Power Requirement for SURCON Central Organization (Example)

Personnel Category / Organization Unit	Computer Eng	Electronics Eng.	Technician	Planning Officer	Adm. Officer	Total Personnel
SURCON Chief	1					1
Main Control Centre Chief	1					1
Operation Branch	7	4			1	12
Maintenance & Support Branch.	4	3	6		3	16
Plan & Project Branch	3	3		1	2	9
SW Development Branch	15				1	16
Procurement & Facilities Branch				4	1	5
Adm. Branch				2	1	3
TOTAL	31	10	6	7	9	63

Such a large organization will have operations and control units deployed all over the country as shown in Fig. 8.1, and an annual expenditure for a country-wide SURCONS organization is estimated to be about 2 million dollars.

9.2.2.5 Public Network Circuit Rental Charges

Some users/owners, particularly the military ones, may use their own private communications networks to meet the needs of their C^3I systems. However, more generally, circuits and networks with appropriate attributes have to be used to interconnect the nodes of the C^3I system and the rental charges for these must, of course, be included in the annual O & M budget.

9.2.3 Total Cost

Investment cost and the operations and maintenance cost inluding expenditures of SURCONS organization and Project Office and public network line charges which are discussed in the previous sections, constitute the total cost of the C^3I system.

By distributing the total system cost to years according to the implementation plan, the annual expenditures in terms of capital and recurring costs can be determined for financial planning.

9.3 IMPLEMENTATION PLAN

9.3.1 Implementation Strategy

Given that the system to be implemented will have open, distributed, secure (preferably MLS) and survivable architecture utilizing wherever possible COTS products and also observing closely the interoperability rules, major factors affecting and governing the implementation process are the difficulty of proper definition of the frequently changing user requirements, rapid advances in the technology and the financial and manpower constraints

The experience gained from the previously implemented C^3I systems for large organizations like the Armed Forces, has dictated the adoption of a step-by-step realization (evolutionary development and acquisition) instead of the system fully implemented at once. These issues are briefly discussed below:

9.3.1.1 Uncertainty in User Requirements

As has already been pointed out, the fundamental issue in a C^3I system is the elicitation of the information requirements, which need to be processed, displayed and exchanged between the users in a timely manner, to enable them to fulfill their missions.

However, the experience shows that, the user requirements, on which the system design is based, are difficult to be defined in appropriate depth, accuracy and

reasonable time, and therefore, are the most problematic, both for the user and the designer alike.

It is often the case that the user has no familiarity with information processing systems and no experience and knowledge about information technologies and therefore is quite unlikely to be able to define his expectations from a C^3I system.

As it has been explained in Sec. 2.5 of Chapter 2 testbedding/prototyping can help the user as well as the system designer/implementer to recognize the problems, needs and expectations from the system and is therefore considered as the first step of the evolutionary development and acquisition process. It is to be noted that this process also enables the system designer to verify the user requirements and determine how these requirements can be satisfied in various ways by modern technology.

9.3.1.2 Technological Developments

Information processing systems are subject to being obsolete fast [9.4] due to rapid technological developments*. Considering also the rapidly changing nature of the user requirements it becomes very difficult to implement complex systems like C^3I system, employing the latest technology if conventional acquisition methods are used.

Evolutionary acquisition approach, however, allows the most effective system establishment through the use of state-of the art technology and takes into account also the changes in user requirements which may emerge at any stage.

9.3.1.3 Financial Constraints

Regardless of the accuracy of the requirements obtained and the technology chosen in the system design, its implementation solely depends on the availability of adequate financial resources. In the conventional acquisition process the implementation of the complete system at once dictates the allocation of the required finds in a short period of time which may not always be possible or desirable.

In summary, the evolutionary acquisition methodology recommended for the implementation of a large scale C^3I system has the following advantages:

- The verification of conformance to the user requirements and the use of suitable technology by means of the testbedding/prototyping approach,
- Realization of the changes in user requirements and evolving technologies prior to full system implementation,

---

* Gordon Moore's law states that chip densities double every other year and according to Bill Joy the speed of microprocessors doubles every 18 months

- System establishment with less financial resource requirements at each stage,
- Implementation of user friendly systems,
- More effective system development to fulfill the requirements,
- Minimum risk and substantial cost saving in the implementation.

Given the above advantages, the evolutionary acquisition methodology does not seem to have any important disadvantages. Frequent update of user requirements, close monitoring of technological developments, careful resource planning and additional efforts for the testbedding/ prototyping development are not considered as disadvantages since all these endeavors will result in the acquisition of a more effective, economic and friendly system for the user.

9.3.2 Principles of Implementation

The major constraints influencing the implementation of a C^3I system are as follows:

- Funding constraints as discussed above,
- Operational priorities as presented by the user,
- Factors related to interoperability with existing and planned systems,
- Requirements to establish progressive improvements in each phase of implementation, while providing a smooth transition to the final configuration,
- The evaluationary introduction of new technology,
- The verification of the proposed architecture through early test/prototype development ,
- The establishment of SURCONS organization and training of personnel,
- Planning of the annual resource allocations for incremental (phased) implementation,
- Rationalization of the architectural design of the system taking into account existing infrastructure and future investments,

In addition, the following factors also affect the implementation of the system:

- Use of COTS products should be preferred when they are available,
- Command Control and conventional Information Management requirements must be met through a common information system in an integrated manner,
- System should be based on reliable and trusted products and subsystems,
- C^3I system should have an Open System Architecture (OSA),
- When implementing OSA, maximum use of international standards should be taken as an objective. In case international standards are not available, commercial and de-facto standards should be adopted.
- In principle all nodes of C^3I system should have similar capabilities,
- The decision as to the use of existing systems in parallel, or to integrate them with the new C^3I system to be implemented, should be made in accordance with cost effectiveness, priorities of operational requirements and compatibility with the system implementation schedule.
- The services required in crisis and war time should be established completely during peace time.
- Every stage of the implementation of the C^3I system should provide certain distinct capabilities to users.
- To avoid high risk applications, verified solutions should be preferred and testbeddig/prototyping approach should be used.
- Standard solutions should be used in every possible situation to ease the interoperability and integration.

9.3.3 Assumptions for Implementation

Following assumptions are made for implementation:

- Majority of the hardware, software and other units in the recommended architectural design is either commercially available or will be in the market in the coming 5 years given the current development rate of technology.
- Simpler applications such as database application programs shall be developed by commercial companies whereas complicated applications such as those related to decision support and data fusion will be prepared by a Project Office established on a continuos basis staffed with appropriate experts.

- The establishment of a project office with full knowledge of system design, is a necessity also for the preparation of bidding documents and evaluation of bids, software checks, system integration and the monitoring of technological developments and standards.

- The ADP personnel to operate, maintain and control of the system would be available and trained prior to the establishment of the system.

- Prior to the preparation of the technical specifications, computation-oriented applications software should be studied in a testbed environment and prototypes should be developed through pilot projects.

9.3.4 Priorities For Implementation

The implementation time table with appropriate phases is prepared by taking into account, the following constraints:

- Operational priorities,
- Financial limitations
- Technological improvements and the internationally accepted standards,
- Software development time,
- System implementation time.

Based on the evaluation and analysis of operational requirements the following operational priorities may be assumed:

i) Fast and reliable information exchange between the nodes,
ii) Rapid and satisfactory information flow among the users within the nodes,
iii) Automatic Data Processing (ADP) support for the node's personnel and activities (common and specific applications),

Functional requirements fulfilling the above operational priorities are:

i) Establishment of Automated Message Handling System organization-wide for rapid and satisfactory information exchange,

ii) Implementation of necessary standards requested for rapid and satisfactory command and control information flow,

iii) Implementation of common ADP support services (LAN, Electronic Mail, Office Automation, Briefing Support, etc.) for the users in all nodes,

iv) Implementation of application programs specific to functional areas for users in all headquarters,

v) Sharing the data, application programs and ADP capabilities between the nodes and users.

Considering all the factors mentioned above we recommend a 3-phase implementation schedule. The pressing user requirements related to Command Control information flow and some common user functions, so-called Core Capability, are implemented in Phase 1. This Phase provides also elements of the architecture (the infrastructure) which are essential to the evolution of the system towards the final configuration. Capabilities called Special Applications which are user specific are implemented in Phase 2. Some of the previous capabilities are extended and decision support which require relatively long time to prototype are implemented in the last phase.

Core Capability will include common applications required by all the users like Message Handling, Electronic Mail, Briefing Support and Office Automation. In order to implement the Open System Architecture within the C^3I system, core capability should be implemented in all the nodes of the C^3I system.

On the other hand; Functional Area Subsystems (Special Applications) will provide the ADP support necessary to carry out command control activities such as intelligence, operations and logistics.

For provision of interoperability between nodes and external systems (National and International), development and implementation of message handling system should have the highest operational priority in order to provide rapid and reliable exchange of structured text and graphics data.

For integrity of the C^3I System, Message Handling System should be implemented first.

Core Capability Package is mainly a procurement activity that has low implementation risk and prototyping is not generally needed because there exists COTS hardware and software products. Part of the already existing management information systems can be integrated to the core capability.

The scope of the implementation phases is reviewed in the following sections.

9.3.5 Implementation Phases

The hardware and the software configurations to be implemented for the C^3I System has been defined in Chapter 6 in detail. These configurations correspond to the final state of the system. As mentioned above it is reasonable to assume that the final C^3I System configuration is reached in three consecutive phases. These are:

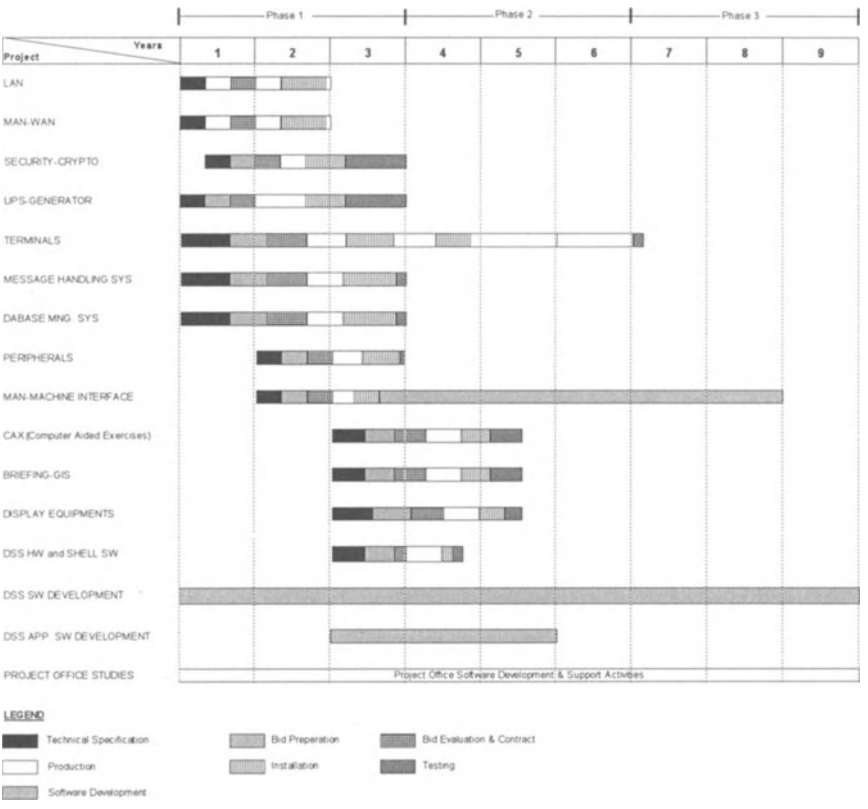
1st Phase	Core Capability	3 years
2nd Phase	Enhanced Configuration	3 years
3rd Phase	Final Configuration	3 years

It is proposed that each phase has an architectural structure that supports and integrates the previous one seamlessly in terms of technical and functional requirements and be implemented after completion of the previous phase.

An example of C^3I system implementation plan consistent with the assumptions made and constraints given in earlier sections, is shown in Table 9.9.

Figure 9.4 shows in shades an example of a node system implementation which considers the three phases, outlined above.

Table 9.9 An Example of a C^3I System Implementation Plan



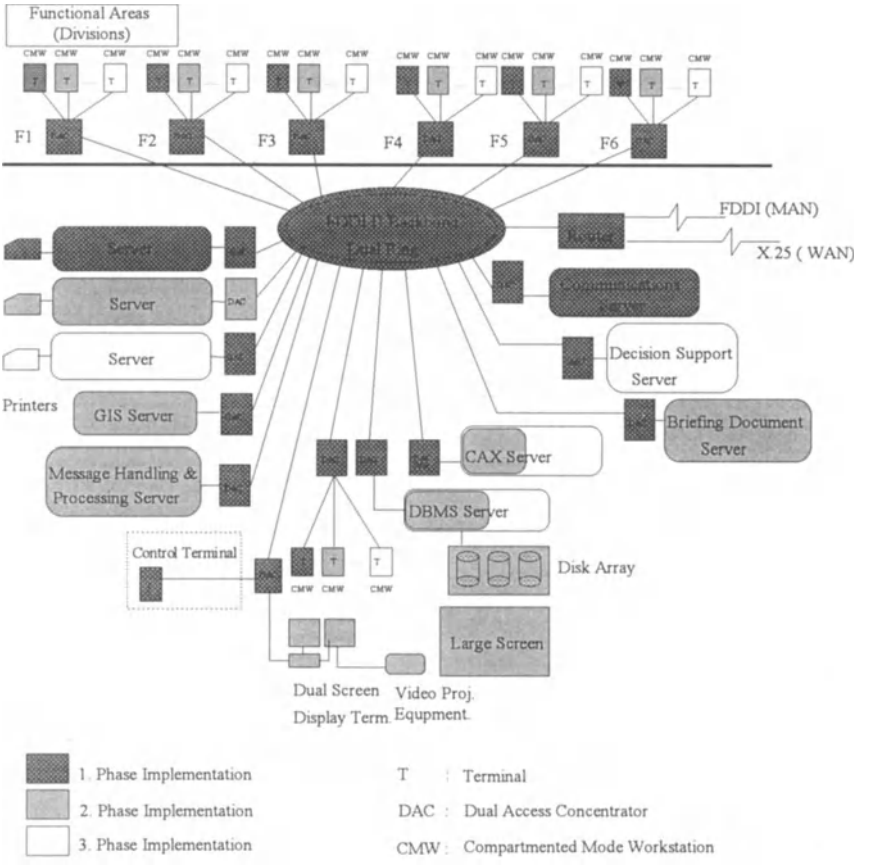


Figure 9.4 An Example of a Node Configuration Including Implementation Phases

9.3.5.1 First Phase, Core Capability

The objectives of this phase are:

- To meet urgent user requirements,
- To implement Automatic Message Handling System,
- To provide elements of the architecture which are essential to the evolution of the system towards the target architecture.

Urgent user requirements are the ADP support for the common user activities and information exchange between and within nodes.

Information exchange between nodes/headquarters is provided by the Automatic Message Handling System (AMHS) [9.4] and this system is augmented by E-Mail service over LANs and WANs. Fundamental system components required to support the AMHS and E-Mail are:

- Message Handling System Server
- User Workstations,
- Local Area Network (LAN),
- Wide Area Network (WAN Gateway),
- Security peripherals such as crypto machines

ADP support for common user activities, in addition to the above include:

- DBMS Server,
- Document Server,
- COTS products providing office automation, briefing support and graphics capability.

LAN and WAN components will constitute the main infrastructure in transition to the final configuration.

It is possible to implement the Core Capability by the available technology with minimum risk and cost in two to three years depending on network size.

The main characteristics of the Core system are:

- Core Capability is based on the "Open System Architecture" concept of the system's final configuration,

- Core Capability can be usually obtained without prototyping and detailed analysis.

- For capability increments requiring testbedding/prototyping, the functional areas can be given as examples in conformance with the evolutionary acquisition principle.

9.3.5.2 Second Phase, Enhanced Configuration

This phase is concerned mainly with special application software development. The software developed in the 1st phase like Graphics-oriented and Database Applications are put into operation in this second phase.

The Database applications for the Functional Area Sub-systems are completed in this phase.

Completed parts of the graphics-oriented applications software become operational in this phase.

First part of the Computation-oriented applications software assumed to start in the first year are completed by the end of this phase.

Computer Aided Exercises, Briefing Support, Geographical Information System and Display Tools are added capabilities of this phase.

Application softwares for the functional areas and common database structures that are developed in different implementation phases are for use organization wide. This ensures vertical and horizontal interoperability between nodes.

It is to be noted that when the application software is developed for each functional area in different phases, Data Dictionary has to be maintained and updated. This activity is strictly followed, directed and controlled by the Data Manager.

Testbedding/Prototyping

As has been pointed out in the previous sections, prototyping/testbedding in the development of the application software that is specific to functional areas (Intelligence, Operations, Logistics) is to be used. This should be undertaken in close coordination with user groups.

Softwares that do not require prototyping should be tested on the "testbed" facilities. Testing, in this respect, is important to verify the system concept and the technical architecture of C^3I system.

Expected Developments in the Second Phase

Development of the database application software for each functional area is carried out without interrupting the system's operation and capabilities obtained in the first phase.

Priority in this phase of special software development will be given to basic activities such as database design, reporting and automatic message processing and preparation. This will ensure the establishment of a sound infrastructure for developing planning and decision support packages which are more complex applications to support the user in the final phase.

Prototype development for Decision Support Applications which started in the first phase will continue in this phase.

Scope of Second Phase

General hardware and software capabilities to be provided in this phase are outlined in Table 9.9.

9.3.5.3 Third Phase, Final Configuration

The objective of this phase is to take the system configuration implemented in the first two phases to its ultimate state in terms of hardware, software, communications, security, interoperability, operation, maintenance, management and control and personnel and training.

The main objective of this phase is to continue to develop and put into operation of the "Computation-Oriented Applications Software" utilizing the results of testbedding/prototyping of the second phase.

Hardware and software to be implemented and the capabilities to be provided in this final phase are shown in Fig. 9.4 and Table 9.9.

9.4 REFERENCES

[9.1] Pressman, R., S., "Software Enginering: A Practitioner's Approach", Mc Graws-Hill Book Co., 1987.

[9.2] Putnam, L., Myers, G, "Measures for Excellence", Prentice Hall, 1992.

[9.3] "Key Technologies for the 21st Century", Scientific American, September 1995.

[9.4] Ince, A. Nejat et al, "Planning and Architectural Design of Integrated Services Digital Networks", Kluwer Academic Publisher, Boston, 1995.

EPILOGUE

Collection, storage, processing and dissemination of information necessary for management and decision making is what C^3I is all about. An ideal system to perform these functions economically and in an easy-to-use manner(user friendly) should have the following features:

- Flexible (ability to change with requirements and technology)
- Reliable (available for a stated time duration)
- Survivable (available under specified stress conditions)
- Secure (need-to-know, resistant to threat)
- Speed (timely information, reasonable response time and user psychology)
- Vendor independent
- Economic to implement and run (use of available COTS software and standards)

The extent to which these features are incorporated into the design of the system depends on:

- How well changing user requirements are captured,
- How well the system is designed,
- Appropriate method of system acquisition
- Availability of software and hardware products

and determines user satisfaction with the system implemented in terms of timeliness of implementation, adaptibility to changing requirements, cost within estimated limits, easy-to-use, operate and maintain and worthwhile improvement in user performance (timely and correct decision making).

The design is driven by the operational requirements some of which are functional (general and user specific) and there are featuers many of which are technical and mostly universal. The more difficult type of knowledge elicitation is the capturing of the functional requirements. These should be stated correctly and fully to allow important design parameters to be derived such as database design, classification of information items, capacity of links and storage devices, speed of processing, transmission and a good reponse time, number and type of terminals, application software categorization (database applications, graphics-oriented applications, computation-oriented applications) and prioritization for implementation, security issues, communication protocols and the standards to be used, information processing topology and finally phasing of implementation and manning, training,

costing and decisions for administrative and organizational issues (e.g. security, data administration).

The method proposed in the book for use in capturing the user requirements is called Mission Oriented Analysis methodology which has been sucessfully used nationally and internationally for well structured establishments such as the military to derive the important system design parameters cited above. This methodology which is described in Chapter 2, can be used also for civilian establishments with some adaptation.

Since the user requirements would be changing continuously in scope and in response to change in technology, there must be an efficient and effective way of ensuring that these changes are reflected fully in the operational system. This can be achieved through the use of CASE tools which are capable of transferring new user requirements into software modifications.

It is the changing requirements rate which may at times exceed the speed of implementation which makes it vital to use evolutionary development and acquisition method with prototyping and this is what has been recommended in the book.

As far as software acquisition is concerned in order to avoid cost and time overruns, use should be made of COTS as much as is consistent with security and interoperability requirements while the user specific software, particularly computation-oriented application software such as decision support aids should be developed in house by a team of specialists the core of whom should be permanantly maintained during the development and operation of the system.

The core capabilities of the system presented in Chapters 3,5 and 6 such as communications, message handling, database management which are prerequisites for the command and control applications are implemented first and before the capabilities of the system are enhanced with application software to be developed. Software cost and time estimation is based on an estimate of the lines of code for every application software involved and the quality of personnel employed in software development. The results are given in statistical terms of the time required and the cost of development as a function of project team capability and size, cost constraints and confidence level. This method, outlined in Chapter 9, is referred to as Quantitative Software Management (QSM) and employs statistical methods and makes use of historical case data.

The order in which software is developed can be established based on functional data models as described in Chapter 6 taking into account interdependencies of software modules.

Having completed logically and fully the more important aspects of requirements and software development as outlined above, the more general system aspects are dealt with as follows. These aspects concern hardware acquisition ,survivability, interoperability, vendor independence, portability and scalability of software and are obtained using Open System Architecture principle and a distributed topology

for the C^3I system. Perhaps the most intractable design issue is the provision of security that is required by the user. It is generally agreed that the best solution is to adopt MLS principles which, however, requires availability of trusted/accredited software with the appropriate security rating. The experience of last years shows that the development, evaluation and accreditation processes of software products take much longer than was originally expected. Given this observation, a feasible strategy to follow, if one cannot wait for the trusted products to be available, would be to use MLS for applications where appropriate software is available such as operating systems and database management systems and for the rest to employ "system high" and "dedicated mode" security policies. The transition from this interim solution to the full MLS would involve replacing the non-trusted software with the trusted counterparts when they become available according to a transition plan. The plan for transition to MLS contains information about the order in which non-trusted software will be replaced and the tasks to be performed such as labelling when porting data to the trusted environment. The details pertaining to the realization of MLS have been presented in Chapter 7.

As far as hardware with required security and reliability is concerned, terminals, servers, storage devices, LANs, WANs, crypto devices, bridges, routers, etc., are available today for use in most C^3I systems which may be implemented readily even though new and emerging information technology may provide smaller, faster, more reliable and hopefully cheaper hardware.

It is hoped that the reader will appreciate that the design methodology described in the book and summarized above, if used by a competent design team, is very likely to provide the user with a C^3I system which meets the stated requirements and remain within the budget limits.

INDEX

A

B

C

D

E

F

M

N

O

P

Q

R

S

W

X