WRDC-TR-90-8007
Volume IV
Part 1

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume IV - IISS System
Part 1 - System Requirements Document

J. Maxwell, M. Foster

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SRD620340000	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. IV, Part 1
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS (City, State, and ZIP Code) 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS (City, State, and ZIP Code) WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL (if applicable) WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 78011F PROJECT NO.: 595600 TASK NO.: F95600 WORK UNIT NO.: 20950607
11. TITLE (Include Security Classification) See Block 19	

12. PERSONAL AUTHOR(S)
Control Data Corporation: Maxwell, J., Foster, M.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr.,Mo.,Day)	15. PAGE COUNT
Final Report	4/1/87-12/31/90	1990 September 30	182

16. SUPPLEMENTARY NOTATION

WRDC/MTI Project Priority 6203

17. COSATI CODES

FIELD	GROUP	SUB GR.
1308	0905	

18. SUBJECT TERMS (Continue on reverse if necessary and identify block no.)

19. ABSTRACT (Continue on reverse if necessary and identify block number)

This document provides the IISS testbed overview and system requirements.

Block 11 - INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Vol IV - IISS System
Part 1 - System Requirements Document

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. (Include Area Code) (513) 255-7371 22c. OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

Table of Contents

Table of Contents

List of Illustrations

List of Tables

SECTION 1

INTRODUCTION AND SYSTEM OVERVIEW

1.1 Background

The objective of the DAPro 6203 project is to support and operate a test bed to validate and demonstrate the concept of Integrated Applications supported by an Integrated Information Support System (IISS). In addition, the project is to support and enhance a set of standards and procedures established by the 6201/2 projects, preceding DAPro, to guide the design of the IISS and to provide guidance to other test bed projects. Finally, a set of requirements is to be established which will be the basis for enhancements to the IISS, as well as, establishing and applying IISS system integration and test and installation standards and procedures.

The activities and accomplishments of the DAPro Project, conducted by Control Data Corporation, are presented in the DAPro Project Final Technical Report, Publication Number FTR620300000, parts 1 and 2, and the Software Availability Bulletin, Publication Number SAB620326000.

1.1.1 Preceding Project Objectives

The objective of the projects preceding DAPro 6203 were intended to provide the test and demonstration vehicle for the ICAM Information Support System concepts described in the 30 September 1981 "Integrated Sheet Metal Center" (Threads Document) and the Project Priority 3101 Computer Based Information System (CBIS) Requirements Document. As the strategy for evolution to the "CBIS data Class II"* and "CBIS data Class I"* environments is developed and implemented, the associated costs and benefits can be tracked against the baseline system. Please refer to the 6201/2 Final Technical Report documents, "Project Overview and Accomplishments", FTR620200000, Vol. I, and "Systems Requirements Document", SRD620240000, Vol. IV, for a complete project history.

1.1.2 Strategy for Evolution

It has been estimated that in large U.S. corporations, most of the existing computer applications will be redesigned over the next 10 to 20 years. It is further expected that, due to the rapidly changing computer technology, the construction techniques and operation modes of new applications will bear little resemblence to those of existing systems.

The Project Priority 3101 CBIS coalition also provided a set of six principles as guides in formulating a solution for the (relatively) near term which are also extensible for the expected long term trends. Individually, each of these principles reflects state-of-the-art technology. These principles are stated as follows:

1. IISS is a key mechanism for the integration of computerized manufacturing. It defines, controls, and executes actions affecting information among various functionally independent subsystems, based on the use of common data.

2. IISS employs a coordinated database approach to support information resource management of various application systems in a closed loop environment within manufacturing.

3. IISS implementation strategy employs several stages of data and application control which allow for increased usage of facilities as management seeks to gain greater benefits from the IISS.

4. IISS operates as a transaction oriented system responding interactively to user commands, rather than to prescheduled batches of computer programs.

5. IISS is accessible from geographically dispersed locations.

These principles and other results of the CBIS project have been taken as a starting point and extended and further articulated by the DAPro Program Office and the Project 6201 coalition. Requir ments, specifications, and the overall system design are being developed with a view of both short and long-term implementation plans for the Test Bed.

The "cost drivers" which the ICAM CBIS Requirements Definition (Project Priority 3101) defined as critically associated with the CBIS environment are summarized in the following nine categories, all of which are being considered as part of the Test Bed IISS design:

1. Data independence - making computer data files independent of the programs which use them.

2. Data nonredundancy - minimizing the number of occurrences of the same data in different files.

3. Data relatability - facilitating the changing of file structure based on specific "views" required by different programs and transactions.

4. Data integrity - improving data quality, consistency, and recoverability.

5. Data accessibility - providing low-cost, user-friendly access to data stored in various files and computers.

6. Data shareability - ensuring that many programs can access the same files simultaneously without degrading performance.

7. Data security - ensuring that data are isolated from users who should not have access to it.

8. Data performance - providing proper controls for changing the CBIS environment over time as changing user needs cause the basic system requirements to change.

9. Data administration - supplying appropriate standards, procedures, and guidelines to ensure consistent evolution of the CBIS environment as demands and technologies change.

Implied by these nine categories are the need for the IISS to operate in a mixed environment containing old and newly developed applications. It is clearly recognized that the existing applications must be supported, while techniques for technology development and new applications development are concurrently provided.

1.2 Summary of Expected Benefits of the Test Bed and IISS

The Test Bed will serve as a step toward realizing the full benefits of a CBIS as represented by the "cost drivers" in the preceding subsection. It also will serve as a facility to assist others to achieve these benefits faster and with less risk. Some of the benefits of the test bed may be summarized as follows:

1. Provide testing facility for individual test bed software products.

2. Demonstrate initial integration of test bed products.

   - Data integration via the Common Data Model

   - Techniques and procedures for more extensive integration of program functions

3. Provide a site for demonstration and evaluation of integrated application products.

   - Applications

   - Methodologies

   - Information support system

4. Reduce risk to subsequent users of integrated application products.

5. Provide standards, guidelines, and procedures.

   - For development of integrated application products
   - For evaluation/adoption by industry

6. Demonstrate strategy for transition from current application processing and development methods to use of the evolving techniques which will subsequently reduce cost and increase system flexibility.

   - Distributed heterogeneous systems, distributed data, and distributed processing.
   - Independence of application data from considerations of actual internal storage organization and database Management System access techniques.
   - Reduced data redundancy.
   - Automated data validation and constraint checking through the Common Data Model.
   - Transaction-oriented applications.
   - Standardized User Interface (similar menu construction for all applications, standard user "HELP" procedures, standard error messages, etc.).
   - Control of execution, in a consistent manner, of processes on different computers using different operating systems.
   - Facilitate and control passing of data and messages between processes on the same or different computers.
   - Consistent error handling throughout the system.
   - System-wide control of startup, shutdown, restart, and recovery.
   - Application programs written using relational database languages referencing nonrelational databases.
   - Independence of application program from the computer on which the user terminal is located.
   - Independence of application program from the characteristics of the terminal on which it will be used.
   - System-supported translation of information formats to host-specific representations.

1.3 Test Bed System Overview

The following is an overview of the 6201/2 hardware and software architecture.

NOTE: The current DAPRO PRoject architecture consists of a VAX 8600 and IBM 4381 interconnected by a local Area Network (LAN).

The hardware architecture described in the following subsection suffices in defining the current Test Bed functionality, as it allows for expansibility and flexibility.

1.3.1 Hardware Architecture

The hardware architecture of the Test Bed supports the interconnection of the heterogeneous computer systems required to demonstrate the functionality of the Test Bed (Figure 1-1).

The Test Bed hardware architecture supports the interconnection of two computer systems via a Local Area Network (LAN) complemented by Wide Area Communication Services.

The two computers embedded in the Test Bed are:

1. A VAX 11/780 computer or equivalent supporting:
    - Test Bed User Interface
    - Test Bed Common Data Model
    - IDSS 2.0 and its database
2. An IBM 3033 computer supporting an MRP package.

The Test Bed makes use of a Local Area Network to interconnect the VAX and the IBM which are in close geographical proximity. This approach offers high throughput, ease of installation, expansion capabilities, and supports the process-to-process communication capabilities required to integrate the heterogeneous databases present in the Test Bed environment.

The Test Bed makes use of Wide Area Communication lines to extend the functionality and usefulness of the Test Bed to computers which are remote geographically.

1. A synchronous leased line provides medium speed communication capabilities to the IBM 3033 located 3.5 miles away from the computer center used to develop the Test Bed.

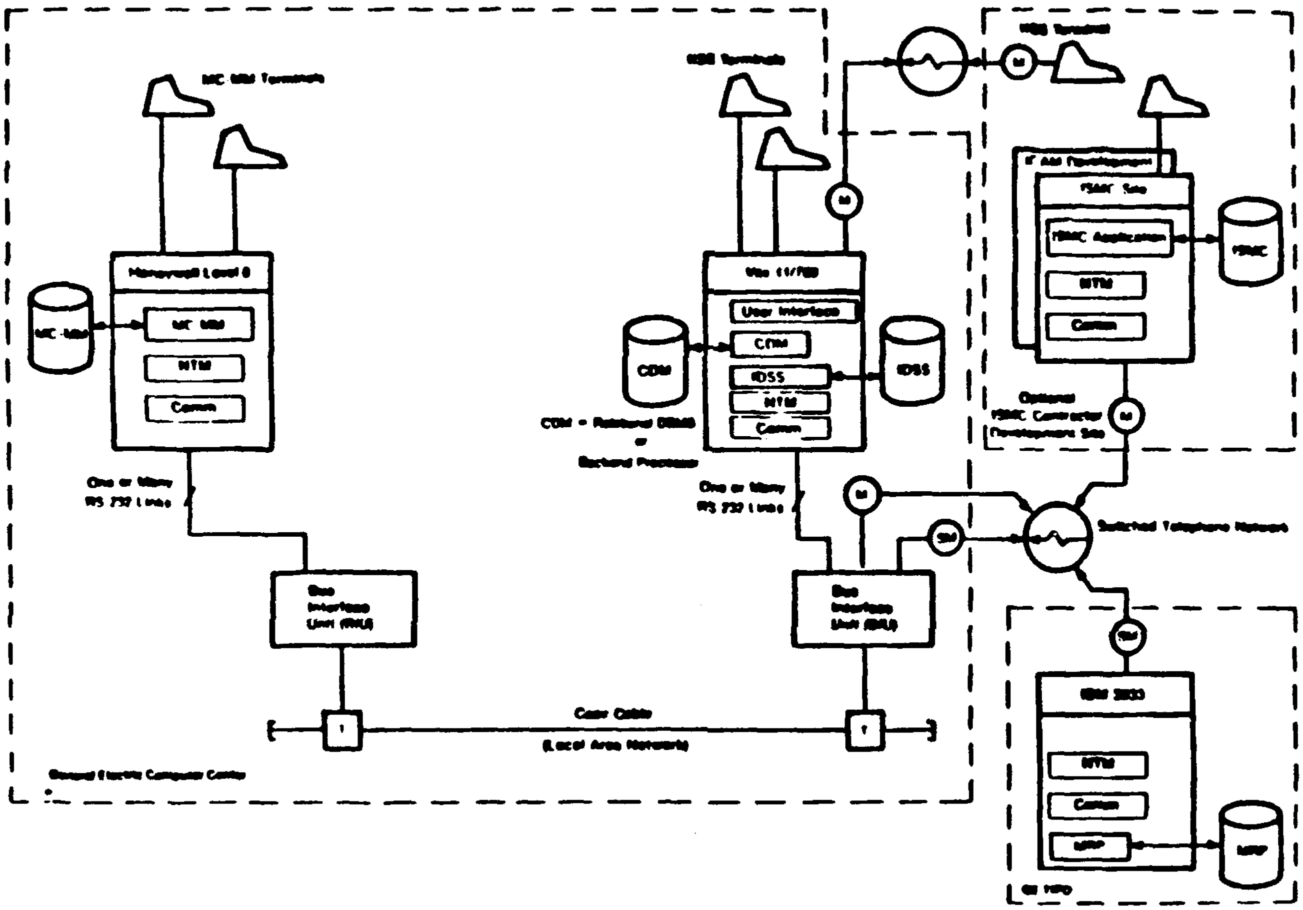

Figure 1-1. Interconnection of Heterogeneous Systems Via Local Area Network

2. Asynchronous lines are provided to interconnect the development computers to the Local Area Network hardware, as well as to interconnect all development terminals to the VAX 11/780 of the Test Bed through the User Interface.

The Test Bed hardware architecture allows for expansibility and flexibility.

1. The Test Bed hardware architecture can be expanded to the MAX Configuration described in the Threads Document. Back end data machines and/or additional general purpose computers can be interconnected via the Local Area Network.

2. The Test Bed hardware architecture can be expanded to a full size production system with minimal changes to the software.

1.3.2 Software Architecture

The major software subsystems also are indicated in Figure 1-1.

1.3.2.1 Distributed Application Data on Heterogeneous Databases

The application data are distributed in heterogeneous database management systems, themselves resident in heterogeneous processors. This approach allows for the integrated query of existing databases by new integrated applications without conversion of the database.

1.3.2.2 Class II Data Integration

The Test Bed software and system utilities support Class II (see subsection 1.1.1) data inquiries. These inquiries may be directed toward any database resident in the Test Bed, and are under the direct control of the Test Bed Common Data Model Processor. System utilities perform data integrity checks selectively on the data being retrieved in the system. The early 1983 implementation of the Test Bed supports updates on the databases bound to the Application Subsystems. Update activities are under the control of the Application Subsystems and are under indirect control of the CDM to the extent that data entry and messages are checked by the CDM. The data integrity checks performed include edit, domain, and range checking. The data required to support the data integrity checks are contained in the Common Data Model, and is under control of the Common Data Model Administrator. Data inquiries use the Test Bed Neutral Data Manipulation Language to query Common Data contained in the databases integrated by the Test Bed. The Test Bed Neutral Data Manipulation Language allows the definition of nonprocedural queries which are independent of the structure of the database(s) being accessed. System services allow the retrieval of data contained in more than one database in more than one system.

NOTE: Many of the software features designated as "future" have been installed by the DAPro Project. Please refer to the Software Availability Bulletin, Publication Number SAB620326000, for a list of all the DAPro-added features and enhancements.

1.3.2.3 User Interface and Data Query via Preplanned Transactions

Information queries via preplanned transactions support the manufacturing scenarios which have been identified and constitute a natural first step toward ad-hoc query. Message integrity checking is supported by system functions, and is performed selectively. The information required to support the message integrity checks is contained in the Common Data Model and is under the control of the Common Data Model Administrator.

1.3.2.4 Integration and Coordination Through the Common Data Model

The Common Data Model is a resource which is maintained in a centralized fashion to support the following functions:

- Define logical structure of the information common to two or more Application Subsystems. The definition includes entities, their attributes, and the relationships between entities.
- Define the domain and values of the entities.
- Access control or authorization information identifying the operations that can be accessed by a particular user. (Future)
- Define the format of the data as stored.
- Catalog of Common database procedures such as schema translation, schema definition, Neutral Data Manipulation Language statement translation, and data translation procedures.
- Locate the specified data in the logical data structure (Test Bed Conceptual Schema).
- Convert query requests to fit the locations of the data and the required processing.
- Aggregate the responses from the various databases.
- Check for the validity and completeness of update requests (Class II environment). (Future)
- Support the user interface.

1.3.2.5 Integration and Coordination Through the Integrated Network Transaction Manager

The Common Data Model provides a repository for the data describing the data, procedures, and policies which are shared between the various IISS Application Subsystems.

The Network Transaction Manager provides the operational implementation of the above concepts, the control of the execution of transactions, the control of the flow of messages through the network, the restart and recovery requirements, and the monitoring of performance.

The Network Transaction Manager is invoked to carry out the following functions:

- o Dispatch of messages through the IISS Network
- o Follow-up on open transactions
- o Logging, time stamping of messages
- o Monitoring of system performance (Future)
- o System synchronization
- o Restart of the IISS system
- o Restart and recovery of the databases (Future)
- o Control of Application Subsystems

The Network Transaction Manager controls the execution of application subsystems by processing the Transaction Message queues built on each node. The queues provide the necessary buffering action resulting from the asynchronous nature of the Test Bed Application Subsystem.

1.3.2.6 Guaranteed Delivery of Messages

The Network Transaction Manager is also invoked to guarantee the delivery of messages. This service is provided to facilitate the migration of the Test Bed to the Class I environment.

This capability guarantees that messages will be delivered, even if the destination application subsystem is temporarily unavailable. To that effect, the messages are uniquely identified at the level of the IISS by journalizing the message type and Application Subsystem of origin, and by time stamping. Time stamping and journalizing allow for the chronological reconstruction of the transaction input stream. This technique supports the recovery of unavailable nodes or Application Processes.

It should be further noted that the system can guarantee that the message was given to the destination process, and even that the process acknowledged having completed processing. The system cannot, however, guarantee that the receiving process actually did process the message and perform the requested functions, or, for that matter, that the message was even read. The proper processing of messages is totally dependent on the receiving application process and cannot be controlled or guaranteed by the IISS system.

1.3.2.7 Standard User Interface

Test Bed User Interface

A Test Bed User Command Language simplifies the task of the user when interacting with the Test Bed. User inputs are through a forms system including menus, formatted data displays, and forms for data entry. The inputs are then formulated into standard messages that are sent to the application processes in the proper host computer.

The User Interface thus provides a unified format to invoke Test Bed System Utilities as well as to support the user dialogues of Application Subsystems specifically designed for the Test Bed.

Virtual Terminal

The proliferation of terminal hardware and the wide disparity in capabilities and features of commercially available terminals create an interfacing problem between IISS and its terminals.

This problem is resolved by defining a specific set of terminal features and protocols which must be supported by the IISS software. This set of features and protocols constitutes the IISS virtual terminal definition.

Specific terminals are then mapped against the IISS virtual terminal software by specific software modules written for each type of real terminal interfaced to IISS. This approach is consistent with the layered software philosophy of IISS since it permits the interfacing of a wide variety of terminals without changes to the IISS application programs.

System-Wide Forms and Protocols

User forms and user protocols are defined at the IISS system level. These forms and protocols define the manner in which the IISS user interfaces with IISS. The forms are data structures with attributes which are enforced by the forms package. Mandatory fields, alphanumeric fields, and numeric only fields are examples of the attributes which are enforced by the forms package.

The forms and protocols are defined by data stored in the CDM, and as such are very flexible and extensible.

1.4 Terms and Abbreviations

Active: Computer enforced (at compile time or at run time).

Activity Framing: Feature which allows to declare a set of Application Processes as being part of a single operation which makes sense from the user viewpoint. All database changes contained within an activity frame are incorporated or else none are incorporated in the databases.

Application Process: A cohesive unit of software that can be initiated as a unit to perform some function or functions.

Application Process Cluster: An Application Process Cluster is the logical grouping of Application Processes and of one Message Processing Unit (MPU) NTM component.

Application Subsystem: An Application Subsystem is composed of one or more application processes and performs specific manufacturing management functions. Instances of Integrated Application Subsystems are: MCMM, IDSS, and a commercially available MRP System.

Class II Data: Data for which query activity is under direct control of the IISS and for which update activity is under indirect control of the IISS.

Common Data

1. Data used by more than one application Subsystem.

2. Data updated by one Application Subsystem and used by another.

3. Data planned to evolve into a category described by (1) or (2) above.

Common Data Model: Describes common data application process formats, screen definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Data Integrity: Improved data quality, consistency and recoverability. The Test Bed common data is subject to the following integrity checks:

1. Type checking

2. Existence checking

3. Edit checking (7 digit telephone number)

4. Attribute value checking (shirtsize = small, medium, large)

5. Range checking

Deadlock: Two processes are said to be dead locked when each process is waiting on the other to complete before proceeding.

Domain Check: Operation which ensures that the values of a given attribute lie within some prescribed set of values. These values may be continuous, discrete, numeric or non-numeric.

Expert/Novice Mode: The User Interface supports the concept of Expert/Novice mode of user interaction. In the novice mode, the user receives tutorial assistance from the system to guide his selection of system features and functions. In the expert mode, the time consuming tutorial assistance is suppressed for maximum efficiency.

Form: Predefined screen format description. The description includes the textual, cursor positioning, data checking information required to display or input data into IISS.

Guaranteed Delivery: Test Bed provided service which ensures the delivery of a message to its destination even if the destination process is unavailable at the time the message is issued.

Integrated (Test Bed) Application Process: An Application Process which:

1. Uses the Neutral Data Manipulation Language to retrieve Class II data which may be distributed on several databases resident on the Test Bed.

2. By the end of the contract, it uses the local database manipulation language to update the local database to which it is bound.

3. Performs its terminal Input/Output operations on the Test Bed terminals.

4. Is controlled from the Test Bed terminals.

Integrated Information Support System: (IISS), a computing environment used to investigate, demonstrate, test the concepts and produce application for information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Non-Integrated (Test Bed) Application Process: An Application Process which:

1. Does not use the Neutral Data Manipulation Language to retrieve Class II data. The local database Management System data manipulation language is used for local database manipulation (update, retrieval).

2. Performs its terminal Input/Output operations on the Test Bed terminals.

3. Is controlled from the Test Bed terminals.

Non Procedural Query: Value stated query. The query statement focuses on what needs to be retrieved rather than on how to carry out the retrieval operations.

<u>Paired Message</u>: A message which contains either one of the following:

a. A request for a reply
   or

b. A reply to a uniquely identified request.

<u>Pre-defined Query Application Processes</u>: Information processing functions implementing predefined data query application processes. The code required for implementation is predefined (manually if necessary) precompiled and linked.

<u>Response Time</u>: Duration of wall clock time between submission of a user request and receipt of the first character of output.

<u>Synchronization Point</u>: Quiet point where the following is true:

1. The Test Bed databases are in a consistent state

2. The state of the queues of pending application process is known and available for future reference

3. The state of the databases is known and available for future reference

4. The state of the queues of pending application process and the state of the databases have been given a common identifier.

<u>System Clean Point</u>: State of the system which satisfies to:

1. The Test Bed databases are in a consistent state.

2. The state of the databases is known and available.

3. The state of the queues of pending messages is known and available.

<u>System Quiet Point</u>: Period of time during which the following is true:

1. The dispatch of messages triggering the execution of application processes is suspended.

2. All dispatched application processes are closed (processing is completed).

3. System quiet points are invoked and terminated under control of the Test Bed operator or Test Bed control mechanism.

<u>Terminal Control Words</u>: A neutral representation of terminal features implemented by specific control characters.

Test Bed Utilities: Test Bed functions that either provide Test Bed operability or facilitate the execution and terminal input/output operations of the Application Processes resident on the Test Bed.

Time to Complete: Duration of wall clock time between submission and completion of a user request.

User Interface : Test Bed services which facilitate the man machine dialogs between the Test Bed services, some of the Integrated or Non-Integrated Application Process resident on the Test Bed and the Test Bed user. The User Interface services are available through the Test Bed terminals.

SECTION 2

TEST BED SYSTEM REQUIREMENTS DOCUMENT

2.1 Scope

2.1.1 Identification

This document defines the concepts, objectives, functions, and other requirements for the DAPro Integrated Information Support System (IISS) Test Bed. This system is intended to be a test computing environment providing integrated data management facilities and distributed processing for heterogeneous databases resident on heterogeneous computer systems interconnected via a Local Area Network.

This document has been prepared by Project Priority 6203 of the Air Force's DAPro Program. This project is being conducted by the Control Data Corporation with the participation of the Software Dynamics Research Corporation, supported by various consultants and contributors.

This project is sponsored by the Manufacturing Technology Division of the Air Force Wright Aeronautical Laboratories.

Please refer to the Software Availability Bulletin, Volume III, Part 16, CI# SAB620326000, for current IISS software and documentation availability.

2.1.2 Background

The requirements stated herein are derived from many sources, including subsystems documents, state-of-the-art technical publication and the authors' experience in the development of computer systems. The CBIS state-of-the-art, Environment, System Requirement Documents, and the various documents transmitted by the DAPro Program Office to the Control Data Corporation are central to the establishment of the present document. These documents are referenced in Section 2.2. This document states the requirements which must be met by the Test Bed.

2.1.3 Functional Description

This document is written to provide:

1. System Requirements that must be satisfied by the IISS Test Bed. These requirements are stated to serve as a basis for mutual understanding between the users, DAPro Program Office personnel, Control Data and the development subcontractors.

2. A basis for system design.

3. A basis for system test.

4. Traceability between system requirements and system needs as expressed in various Air Force documents.

2.2 References

2.2.1 Applicable Documents

Systran, Inc., ICAM Documentation Standards, ICAM Document IDS 150120000A, 28 Dec 81.

A.D. Little, ICAM Computer Based Information System (CBIS), System Environment Document, ICAM Document SED 310140000, Sept 81.

A.D. Little, ICAM Computer Based Information System (CBIS), State of the Art Document, ICAM Document SAD 310140000, Sept 81.

A.D. Little, ICAM Computer Based Information System (CBIS), System Requirement Document, ICAM Document SRD 310140000, Sept 81.

ICAM Program Office, The Integrated Sheet Metal Center, 30 Sept 81.

N. Tupper, Memorandum for the Record, 5 Oct 81.

2.2.2 Terms and Abbreviations

See Section 1.4 of this document.

2.3 Requirements

2.3.1 Specific Requirements

2.3.1.1 Specific Requirements and Constraint Summary

Functional requirements for the IISS Test Bed are derived from a needs analysis attached to this document as Appendix A. Table 2-1 on the following pages is a succinct list of these requirements. The table columns are explained in the "Notations" section of the table.

TABLE 2-1

FUNCTIONAL REQUIREMENTS

	FUNCTIONS	NEEDS
R1	Provide Data Shareability Between Integrated and Non-Integrated Application Subsystems	Needs A.3.1
R2	Provide Common User Interface to Integrated and Non-Integrated Applications Subsystems	Needs A.3.7
R3	Measure and Report System Performance	Needs A.3.11
R4	Provide Test Bed System Control	Needs A.3.3 Needs A.3.6
R5	Maintain Test Bed System Control Information	Engineering Judgment
R6	Provide Capability to Implement and Test Programs on Test Bed	Engineering Judgment

	SYSTEM CONSTRAINTS	NEEDS
SC1	General System Constraints	Needs A.3.12 Needs A.4.1 Needs A.4.2 Needs A.5.0

Notation:

1. Each requirement is identified by a unique requirement number (Example R3).
2. Each system constraint number is uniquely identified by a unique system constraint number (Example SC1).
3. Each constraint is uniquely identified by its constraint number within the scope of the requirement to which it applies (Example C3.2).
4. The above table cross references Requirements and System Constraints to the Specific Needs shown in Appendix A, or to accepted Engineering principles.

Table 2-2 lists all constraints bearing on the requirements identified in Table 2-1. The remainder of this subsection is a description of the requirements and constraints identified in Table 2-1 and 2-2.

TABLE 2-2

R1: DATA SHAREABILITY BETWEEN INTEGRATED AND NON-INTEGRATED APPLICATION SUBSYSTEMS

	CONSTRAINTS	NEEDS
	PERFORMANCE	
C.1.0	Query Capabilities	Engineering Judgment
C.1.01	Dead Lock Resolution	Engineering Judgment
	PHYSICAL	
C.1.1	Heterogeneous Computer Systems	Needs A.3.5.1
C.1.2	Heterogeneous Database Managers	Needs A.3.4.1
C.1.3	CODASYL Databases	Needs A.3.4.2
	TEST	
C.1.4	Traceable Messages	Engineering Judgment
	DESIGN	
C.1.5	Query Transactions	Engineering Judgment
C.1.5.1	Integrated Application Processes	Needs A.3.9.1
C.1.6	Migration to Class I Integration	CBIS A.2.6.4
C.1.7	Class II Data Integrity Checking	Engineering Judgment
C.1.7.1	Message Integrity Checking	Engineering Judgment
C.1.8	Minimize Modifications to Host Operating Systems	Engineering Judgment
C.1.9	Minimize Modifications to Existing Application Systems	Engineering Judgment
C.1.10	Transparent Data Location	Needs A.3.6.3
C.1.11	Transaction Processing	Needs A.3.9.3
C.1.12	Prioritized Transaction Processing	Needs A.3.9.4
C.1.13	Predefined Transaction Messages (Early Implementation)	Engineering Judgment
C.1.14	Active Data Security	Needs A.3.3.7

TABLE 2-2 (Cont'd)

R2: PROVIDE COMMON USER INTERFACE TO INTEGRATED AND NON-INTEGRATED APPLICATION SUBSYSTEMS

	CONSTRAINTS	NEEDS
	PHYSICAL	
C.2.0	Single Terminal Interface	Engineering Judgment
C.2.1	Virtual Terminal Interface	Needs A.4.2.4
	DESIGN	
C.2.2	Consistent Invocation and Control of Application Subsystems	Needs A.3.7.1
C.2.3	Invocation and Control of Test Bed Utilities	Needs A.3.7.4
C.2.4	Extensibility, Flexibility of User Interface	Needs A.3.7.2
C.2.5	Application Subsystems Security	Needs A.3.6.2
C.2.6	Centralized User Interface Support Data	Needs A.3.6.8
C.2.7	Forms Generator	Needs A.3.7.3
C.2.8	Report Generator	Needs A.3.7.3

R3: MEASURE AND REPORT SYSTEM PERFORMANCE

	CONSTRAINTS	NEEDS
	PERFORMANCE	
C.3.1	Measure Elapsed Time and Response Time Within the Time Resolution of the Host Operating Systems	Engineering Judgment
C.3.2	Measure Transaction Frequency, Elapsed Time, Response Time, and Provide Selective Reporting	Needs A.3.11
C.3.3	Measure Test Bed Resource Usage and Provide Selective Reporting	Needs A.3.11
C.3.4	Use vendor Supported Resource Monitors	Engineering Judgment
C.3.5	Support User Accountability	

TABLE 2-2 (Cont'd)

R4: PROVIDE TEST BED SYSTEM CONTROL

	CONSTRAINTS	NEEDS
C.4.1	Provide Communication Between Application Processes	Engineering Judgment
C.4.2	Provide Active Routine of Transaction Messages	Needs A.3.6.3
C.4.3	Control Execution of Application Processes	Engineering Judgment
C.4.4	Provide Error Processing	Engineering Judgment
C.4.5	Maintain Test Bed Operability	Needs A.3.6.1
C.4.6	Support Test Bed Recovery	Needs A.3.4.3
C.4.7	Provide Message Guaranteed Delivery Service	USAF Verbal Request

R5: MAINTAIN TEST BED SYSTEM CONTROL INFORMATION

	CONSTRAINTS	NEEDS
	PHYSICAL	
C.5.1	Centralized System and Common Data Control	CBIS A.1.4
C.5.2	Controlled Access of System and Common Data Control Information	CBIS A.1.13
C.5.3	Flexibility of the Common Data Control Structure	Engineering Judgment
C.5.4	Test Bed System Control Migration	Needs A.4.2.5
C.5.5	The Maintenance of the Test Bed Control Information Is Computer Assisted	Engineering Judgment
	DESIGN	
C.5.6	A Common Data Model Supports the System and Common Data Control Information	CBIS A.1.1
C.5.7	Selective Access to the Maintenance Functions	Engineering

TABLE 2-2 (Cont'd)

R6: PROVIDE CAPABILITY TO IMPLEMENT AND TEST PROGRAMS ON TEST BED

	CONSTRAINTS	NEEDS
C.6.1	Protect System from New Application Programs	Engineering Judgment
C.6.2	Provide Real Data for Testing	Engineering Judgment
C.6.3	Make Maximum Use of Vendor Supplied Utilities	Engineering Judgment
C.6.4	Allow Analysis of Message Journals	Engineering Judgment
C.6.5	Test Bed May Be Shut Down for Special Conditions	Engineering Judgment
C.6.6	Test Bed Shall Be Accessible	Engineering Judgment
C.6.7	Tests Shall Be Repeatable	Engineering Judgment
C.6.8	Test Protection Modes Shall Be Easily Changeable	Engineering Judgment

TABLE 2-2 (Cont'd)

	GENERAL SYSTEM CONSTRAINTS	NEEDS
	PERFORMANCE	
SC1.1	Engineering Trade-offs Consider Impacts on Performance	Engineering Judgment
SC1.2	Test Bed Design Is Expansible	Needs A.3.5.5
SC1.3	Test Bed Performance Improvements	Engineering Judgment
	PHYSICAL	
SC1.4	Heterogeneous Hardware Environment	Hardware Availability
SC1.5	Geographic Constraint	Hardware Availability
SC1.6	Local Area Network Requirements	Needs A.3.5.3 Needs A.3.5.4 Needs A.3.5.5
SC1.7	Heterogeneous Database Management Systems	Availability Engineering
SC1.8	Test Bed Terminals	Engineering Judgment
SC1.9	Technology Transfer by 1 Feb 83	Needs A.5.1
SC1.10	Test Bed Hardware Is Expansible	Engineering Judgment
	TEST	
SC1.11	Test Configuration	Engineering Judgment
SC1.12	Modular Testing, Phased Integration	Engineering Judgment
SC1.13	Demonstration Class II Integration	
SC1.14	MC-MM, IDSS 2.0, and MRP Demonstration	Needs A.5.2 USAF Verbal Request
SC1.15	MRP Selection Criterion	USAF Verbal Request

TABLE 2-2 (Cont'd)

	GENERAL SYSTEM CONSTRAINTS	NEEDS
	DESIGN	
SC1.16	The System and Common Data Control Promotes Integration Through Sharable databases	Engineering Judgment
SC1.17	The Test Bed Architecture Is Layered and Modular	Needs A.3.12.1 Needs A.3.12.3 Needs A.4.1.5
SC1.18	The Test Bed Software Adheres to the Test Bed Standard Guidelines and Procedures	Engineering Judgment
SC1.19	The Test Bed Software Is Extensible and Flexible	CBIS B.3
SC1.20	Physical and Logical Data Structures Are Decoupled	Needs A.4.2.1
SC1.21	The Test Bed Software Is Portable	Needs A.3.3.2
SC1.22	Binary Data Transfer	Needs A.3.2.4

Table 2-3

SYSTEM REQUIREMENT DOCUMENT AND
NEEDS ANALYSIS CROSS REFERENCE TABLE

NEEDS STATEMENT	SRD LINE ITEM
A.3.0 Test Bed NEEDS	
A.3.1 Integration Needs	R1
A.3.1.1 Integration Demonstration Needs	SC1.13
A.3.1.2 Integration Definition	Glossary
A.3.1.3 Integration Scope	
A.3.1.4 Excessive Data Redundancy	
A.3.2 Interfacing Needs	
A.3.2.1 Interfacing Stand Alone CAD&CAM Systems	
A.3.2.2 Interfacing Standards	
A.3.2.3 Interface to Process Planning	
A.3.2.4 Machine to Machine Interface Problems	SC1.21
A.3.3 Data Management Needs	R5
A.3.3.1 Data Structuring Needs	C5.6
A.3.3.2 Data Portability Needs	----
A.3.3.3 Data Validation Needs	C1.7, C1.7.1
A.3.3.4 Data Integrity Needs	C1.7, C1.7.1
A.3.3.5 Data Redundancy Elimination Needs	Consequence of C5.6
A.3.3.6 Time Dependency of Data Needs	----
A.3.3.7 Data Security Needs	C1.14
A.3.3.8 Active Data Dictionary Needs	----
A.3.4 Distributed Database Needs	
A.3.4.1 Distributed Database Needs	C1.2
A.3.4.2 database Codasyl Compatibility Needs	C1.3
A.3.4.3 Individual Database Recovery and Backup Needs	C4.6
A.3.5 Heterogeneous Hardware Needs	
A.3.5.1 Multiprocessors Heterogeneous Hardware Needs	C1.1

Table 2-3

SYSTEM REQUIREMENT DOCUMENT AND
NEEDS ANALYSIS CROSS REFERENCE TABLE (Cont'd)

NEEDS STATEMENT		SRD LINE ITEM
A.3.5.2	Back End Database Machine Needs	C5.4
A.3.5.3	Local Area Network Needs	SC1.6
A.3.5.4	Local Area NetworK Standardization and Planning Needs	SC1.6
A.3.5.5	Local Area Network Expansibility Needs	SC1.2
A.3.6	Common Data Control Needs	R5
A.3.6.1	System Usable Directory of Network Characteristics Needs	C5.1
A.3.6.2	System Usable Security Directory Need	C2.5
A.3.6.3	System Usable Location Directory Needs	C1.10; C4.2
A.3.6.4	Comprehensive Data Dictionary Needs	----
A.3.6.5	Active Data Dictionary Needs	----
A.3.6.6	ISMC Common Data Needs (Min)	----
A.3.6.7	Common Data Definition Needs	Glossary
A.3.6.8	ISMC Common Data Needs (Max)	C2.6
A.3.6.9	ISMC Improved Constraint Checking of Common Data (Max)	C1.5
A.3.7	User Interface Needs	R2
A.3.7.1	User Interface Consistency Needs	C2.2
A.3.7.2	User Interface Flexibility Needs	C2.4
A.3.7.3	Standard Query/Report Generator Needs (Min)	C2.7; C2.8
A.3.7.4	User Command Language Needs (Mid 82)	C2.3
A.3.7.5	Standard User Interface Needs (Mid)	C2.2
A.3.7.6	User Ad-Hoc Queries Needs (Min)	----

Table 2-3

SYSTEM REQUIREMENT DOCUMENT AND
NEEDS ANALYSIS CROSS REFERENCE TABLE (Cont'd)

NEEDS STATEMENT	SRD LINE ITEM
A.3.8 Simulation Needs	
A.3.8.1 Manufacturing Planning Simulation Needs	Provided by IDSS 2.0 (Not Test Bed Requirement)
A.3.8.2 Master Schedule Planning Simulation Needs	
A.3.8.3 Simulation Batch-Run Needs	
A.3.8.4 Simulation Implementation Needs: IDSS	
A.3.9 Processing Needs	
A.3.9.1 database Updating Needs (Min)	----
A.3.9.2 database Updating Needs (Mid)	----
A.3.9.3 Transaction and Batch Processing Needs (Min)	C1.11
A.3.9.4 Transaction Prioritization Needs	C1.12
A.3.9.5 Processor Load Leveling Needs (Max)	----
A.3.10 Response Time Needs	
A.3.10.1 Immediate Retrieval of Engineering Changes Needs	Not Applicable to Demonstration Test Bed
A.3.10.2 Eight Hour Retrieval of Engineering Change Preparation Data	
A.3.11 Test Bed Resource Usage Statistics Needs	R3
A.3.12 Test Bed Standards and Procedures Needs	----
A.3.12.1 Test Bed Development Standard Needs	SC1.18
A.3.12.2 Test Bed Application Standard Needs	SC1.18
A.3.12.3 Test Bed Layering Standard Needs	SC1.17

Table 2-3

SYSTEM REQUIREMENT DOCUMENT AND
NEEDS ANALYSIS CROSS REFERENCE TABLE (Cont'd)

NEEDS STATEMENT	SRD LINE ITEM
A.4.0 PROJECT DRIVERS	
A.4.1 Responsiveness to Change Needs	SC1.19
A.4.1.1 Responsiveness to Changes in Application Needs	
A.4.1.2 Responsiveness to Changes in Data Needs	C1.5
A.4.1.3 Responsiveness to Changes in Equipment and Technology Needs	SC1.19
A.4.1.4 Responsiveness to New Data Processing Technology Needs	SC1.19
A.4.1.5 Layered Architecture Needs	SC1.17
A.4.2 Portability Needs	
A.4.2.1 Decoupling Logical and Physical Data Structures Needs	SC1.20
A.4.2.2 Eliminating Program Dependency on Peripherals Needs	----
A.4.2.3 Eliminating Dependency on Vendor Data Systems Needs	----
A.4.2.4 Virtual Terminal Needs (Mid)	C2.1
A.4.2.5 Migrating IISS Dictionary to Back End Processor Needs	C5.4
A.4.3 Technology Development Needs	
A.4.3.1 Systematic Development of Integration Technology Needs	Project Management Not System Requirement
A.4.3.2 Evolutionary Technology Development Needs	
A.5.0 PROJECT MANAGEMENT NEEDS	
A.5.1 Schedule Needs	SC1.9
A.5.2 Test Bed Configuration Needs: MRP and IDSS	SC1.14; SC1.15
A.5.3 Test Bed Expected Life Needs	Project Management Not System Requirement
A.5.4 Project Traceability Needs	
A.5.5 Test Bed Affordability Needs	
A.5.6 Technology Transfer Needs	SC1.9

The above Needs Analysis cross reference table show the relationship existing between the statements of Needs taken from the Needs Analysis Document and the Requirements, Constraints and System Constraints shown in the System Requirement Document (SRD).

2.3.1.2 Cross Reference of Needs and Requirements

This subsection contains a cross reference of the Test Bed needs (see Appendix A) and of the specific requirements and constraints identified in Section 2.3.1.1. The cross reference list is given in Table 2-3.

2.3.1.3 Specific Requirement and Constraint Statements

This subsection contain a description of the specific requirements and constraints identified in Section 2.3.1.1.

o (R1) REQUIREMENT 1: Provide data shareability between integrated and non-integrated Application Subsystems.

Description: The Test Bed provides the integration environment for integrated and non-integrated Application Subsystems. In this environment, data items which are used by more than one Application Subsystem may be shared. Access to common data is performed via Test Bed system services.

Requirement 1 is subject to the following constraints:

PERFORMANCE

C1.0 - Query Capabilities
Relational database oriented (one or more records rather than one record at a time) query capabilities are provided to support new applications specifically designed to be integrated on the Test Bed.

C1.0.1 - Deadlock Resolution
The Test Bed resolves Application Subsystem deadlocks over shared system resources. (Future)

PHYSICAL CONSTRAINTS

C1.1 - Heterogeneous Computer Systems
The Test Bed is implemented by interconnecting several distinct and dissimilar computer systems, each running under commercially available operating system.

C1.2 - Heterogeneous Database Managers
The application subsystems integrated by the test bed are supported by commercially available database management systems. Since the application subsystems already exist, these database management systems are, in general, dissimilar.

C1.3 - Databases Structures

1. The database managers which support the application subsystems to be integrated by the test bed are compatible with CODASYL DDLC specifications.
2. The design used to support CODASYL DDLC database managers must be expansible to:
   A. Hierarchical Database Managers
   B. Index Sequential Files

   Note: Paragraph 2 results from a verbal request from the DAPro program office.

Test

C1.4 - Traceable Messages
Tracing of messages is supported to facilitate the debugging and testing of the Test Bed software.

DESIGN

C1.5 - Query Transactions
Query transactions use the Test Bed Neutral Data Manipulation Language (NDML) to query Common Data contained in the databases integrated by the Test Bed. These query transactions allow access to Class II data.

C1.5.1 - Integrated Application Processes
Integrated application processes use query transactions to query Common Data. Updates are only performed on the database bound to the application. These updates are performed without supervision of the Common Data Model.

C1.6 - Migration To Class I Integration
The Early Implementation Test Bed is restricted to Class II data integration to satisfy the schedule and dollar constraint of the project. However, enhancements to the Test Bed will strive to include Class I data integration. The Early Implementation design must be compatible with a Test Bed Class I design.

C1.7 - Class II Data Integrity Checking
The data integrity checks are performed selectively on data being updated or retrieved. In test mode, these checks are performed in a mandatory fashion. (Future)

C1.7.1 - Message Integrity Checking
Message integrity checks are performed selectively on messages exchanged in the Test Bed. These checks include edit, domain and range checking of the information conveyed in the message. (Future)

C1.8 - Minimize Modifications to Host Operating Systems
The design of the Test Bed minimizes the number and the complexity of the changes required to the operating systems of the hosts computers. It is desirable that such modifications be avoided, if at all possible.

C1.9 - Minimize Modifications to Existing Application Subsystems
The design of the Test Bed minimizes the number and the complexity of the changes required to install existing Application Subsystems. Existing Application Subsystems may be modified and recompiled to facilitate installation.

C1.10 - Transparent Data Location
A system supported procedure is used by the new application processes to retrieve data distributed in the Test Bed databases. This procedure is not dependent upon the location of the data, as viewed from the application process.

C1.11 - Transaction Processing
The Early Implementation Test Bed provides transaction processing. This implementation is restricted to preplanned application processes.

C1.12 - Prioritized Transaction Processing
Application processes are prioritized by schedule time of execution and by relative importance for those scheduled to run at the same time. The priority information is associated with in the message initiating the application process. (Future)

C1.13 - Predefined Transaction Messages
The Early Implementation Test Bed relies on predefined messages to control the execution (initiate, abort) of predefined application processes. The format of the predefined messages is compatible with the format of messages to be defined in the message definition language which will be developed as enhancement to the Early Implementation Test Bed. (Future)

C1.14 - Active Data Security
For the ad-hoc environment, the access to common data is discriminatory, and is dependent upon the access privileges of the user. For predefined, the application process environment access to common data is granted to specific application processes. (Future)

- (R2) REQUIREMENT 2 - Provide Common User Interface to Integrated and Non-Integrated Applications Subsystems

Description: a Common User Interface to the integrated and non-integrated Application Subsystems and Test Bed utilities is provided.

Requirement 2 is subject to the following constraints:

PHYSICAL

C2.0 - Common Terminal Interface
A common terminal interface is used to access the various Application Subsystems and Test Bed utilities.

C2.1 - Virtual Terminal Interface
A standard set of terminal features and formats is defined for the Test Bed. The conversion of these features and formats to and from the specific features and formats of a given terminal hardware or Application Subsystem is performed by the Virtual Terminal Interface.

DESIGN

C2.2 - Consistent Invocation and Control of Application Subsystems
The User Interface provides the ability to invoke and to control the Application Subsystems included on the Test Bed. The User Interface does not depend upon the host on which the Application Subsystem reside. The User Interface is only available to the Test Bed users.

C2.3 - Invocation and Control of Test Bed Utilities
The User Interface provides the ability to invoke and to control the utilities provided by the Test Bed System (user help facility, menu selection program, etc.).

C2.4 - Extensibility, Flexibility of User Interface
The User Interface is extensible and flexible. It allows new functions and new hosts to be supported.

C2.5 - Application Subsystem Security
The User Interface controls the access of users to the various Test Bed Application Subsystems. The information required to support this function is provided by the Test Bed system control and common data description repository.

C2.6 - Centralized User Interface Support Data
The User Interface support data (forms, error messages, etc.) is centrally controlled to improve User Interface consistency. (Partial)

C2.7 - Forms Generator
A general purpose forms generator is provided to enhance the flexibility of the User Interface for new Application Subsystems and Test Bed utilities.

C2.8 - Report Generator
A general purpose report generator is provided to enhance the flexibility of the User Interface for new Application Subsystems. The report generator is not included in the Early Implementation Test Bed. (Provided in Release 2.0)

o (R3) REQUIREMENT 3 - Measure and Report System Performance (Future)

Description: the Test Bed resource usage and transaction frequency and performance are monitored to support the objective assessment of the Test Bed.

Requirement 3 is subject to the following constraints:

PERFORMANCE

C3.1 - Measure Elapsed Time and Response Time Within the Time Resolution of the Host Operating Systems

C3.2 - Measure Transaction Frequency, Elapsed Time, Response Time, and Provide Selective Reporting
User and system transaction frequency, elapsed time, and response time will be measured. Totals, means and standard deviations of the transaction measures are selectively provided upon request.

C3.3 - Measure Test Bed Resource Usage, and Provide Selective Reporting
Measure the usage of the Test Bed hardware and software resources. These measurements include the usage of the Test Bed hosts and of the Test Bed local area network. Totals, means and standard deviations of these measurements are selectively provided upon request.

C3.4 - Use Vendor Supported Resources Monitors
Resource usage monitors provided by the various vendors are used to the maximum extent possible to reduce system overhead.

C3.5 - Support User Accountability
An audit trail is kept on users and data that are entered.

o (R4) REQUIREMENT 4 - Provide Test Bed System Control

Description: 1) Control the Test Bed system
2) Maintain operability

Requirement 4 is subject to the following constraints:

PHYSICAL

C4.1 - Provide Communication Between Application Processes The Test Bed provides communication services to facilitate communication between Application Processes and other Test Bed software.

C4.2 - Provide Active Routing of Transaction Messages
Transaction messages are automatically routed in the Test Bed. The information required to route the

transaction messages is supplied by the system control information repository.

C4.3 - Control Execution of Application Processes
The execution of Application Processes is controlled by transaction messages. These messages are used to initiate, to abort and to redirect the I/O of new or existing Application Processes.

C4.4 - Provide Error Processing

1. Detect and log the errors originating in

   a. Transmission of messages

   b. Retrieval and manipulation of system and Common Data control information

   c. Execution of any program part of the Test Bed system

2. Support the error processing capabilities provided by the Application Processes prior to their installation.

C4.5 - Maintain Test Bed Operability

o Test Bed start-up

o Test Bed shut down

o Test Bed host restart/stop

o Test Bed back-up (Future)

o Test Bed Subsystem (cf: Local Area Network, etc.) restart/stop

C4.6 - Support Test Bed Recovery (Future)

1. Automatic recovery and synchronization of databases following abnormal termination of Application Processes, to the extent supported by the local database managers.

2. Manual recovery and synchronization of databases for other abnormal operating conditions (crash, etc.), to the extent supported by the local database managers.

C4.7 - Provide Message Guarantee Delivery Service

1. The delivery of Test Bed Messages may be guaranteed to occur, even though the destination process is not available at the time the message is generated.

2. Even though the delivery of the message may be guaranteed, no constraint is placed on the timing of the delivery.

- (R5) REQUIREMENT 5 - Maintain Test Bed System Control Information

Description: The information supporting the active Test Bed system and common data control must be maintained. Maintenance functions include the creation, deletion, updating and retrieval functions.

Requirement 5 is subject to the following constraints:

PHYSICAL

C5.1 - Centralized System and Common Data Control
The information required to control System (Future) and Common Data is centrally controlled.

C5.2 - Controlled Access of System and Common Data Control Information
The description of the common data is protected, and is supplied to authorized users only.

C5.3 - Flexibility of the Common Data Control Structure
The Common Data Control Structure can describe a wide variety of different data structures. It can be extended without disrupting Test Bed services. (Partial)

C5.4 - Test Bed System Control Migration
The structure of the Test Bed system control information allows migration to a back end data machine.

C5.5 - The Maintenance of the Test Bed Control Information Is Computer Assisted
Computer supported functions assist the user in creating, deleting, updating, retrieving and checking the information used to support the control of the Test Bed System and Common Data.

DESIGN

C5.6 - A Common Data Model Supports the System and Common Data Control Information
A complete description of the common data is required to achieve integration. This description is supported by a well defined structure that allows for the description of:

1. Entity classes and the relations among them

2. Mappings between entity classes and their attribute classes

3. Constraints which define permissible values for attributes

C5.7 - Selective Access to the Maintenance Functions
Access to the system control information maintenance functions is provided to authorized users only.

o (R6) REQUIREMENT 6 - Provide Capability to Implement and Test Programs on Test Bed

Description: the Test Bed should provide an environment to allow new programs to be tested and installed into the total system.

C6.1 - Protect System from New Application Programs (Future)
During testing, the new application programs should be prevented from making unauthorized changes to the system including the system data.

C6.2 - Provide Real Data for Testing
A capability shall be provided to allow real data to be used during testing, but to prevent modifying any protected system data or Application Subsystem data. (Protection = Future)

C6.3 - Make Maximum Use of Vendor Supplied Utilities
The vendor supplied utilities should be used where possible for development and test. Actual development will be host dependent, but provisions must be made to introduce and use the Test Bed capabilities as the program is integrated into the total system.

C6.4 - Allow Analysis of Message Journals
A capability will be provided to allow analysis of message journals for messages affecting a given program.

C6.5 - Test Bed May Be Shutdown for Special Conditions
The Test Bed is not required to run continuously, and can be shutdown under special conditions.

C6.6 - Test Results Shall Be Accessible
The results of calculations and data updates made during testing shall be available to the programmer even if the results are not permanently stored.

C6.7 - Tests Shall Be Repeatable
It shall be possible to repeat tests with known test data.

C6.8 - Test Protection Modes Shall Be Easily Changeable
The degree of protection of a program and the data it references shall be easily changeable, though with proper access security to provide reasonable protection for the total system. (Partial)

- SC1 - GENERAL SYSTEM CONSTRAINTS
  The Test Bed must, in addition, satisfy the following constraints.

PERFORMANCE

SC1.1 - Engineering Trade-Offs Consider Impact on Performance
The Test Bed is experimental in nature, and its expected level of performance cannot be quantified. Careful design trade-offs are made to minimize user response and elapsed time.

SC1.2 - Test Bed Design Is Expansible
The Test Bed design allows reasonable expansion of the number of users and Application Subsystems to be accommodated, subject to environmental constraints.

SC1.3 - Test Bed Performance Improvements
The Test Bed design allows improving performance through reconfiguration of hardware and software.

PHYSICAL

SC1.4 - Heterogeneous Hardware Environment
The Early Implementation Test Bed is implemented by interconnecting:

1. A Vax 11/780 under VMS

2. A Honeywell Level 6 under GCOS6/MOD400 OS

3. An IBM 3033 under MVS

SC1.5 - Geographic Constraint
The Vax 11/780 and the Honeywell Level 6 are located in Building 4 of the General Electric Schenectady complex. The IBM 3033 is located at the General Electric CIPO facility. (GE's main facility later moved to Albany, NY, and IBM support to GE/Rockville, Md., and then to Boeing/Wichita.)

SC1.6 - Local Area Network Requirements
A standard, commercially available, Local Area Network system is used to interconnect the Test Bed host computers. Owing to the geographic constraint bearing on the Early Implementation Test Bed, wide area services are required to interconnect the IBM 3033 to the local area network.

SC1.7 - Heterogeneous database Management Systems
The Early Implementation Test Bed integrates Application Subsystems or appropriate test programs supported by the following database management systems:

1. Vax 11/780 : DBMS and ORACLE

3. IBM 3033 : Codasyl DBMS Supporting Selected MRP; extended to IDMS (Cullinet), TOTAL, and IMS.

[Later extended to VAX-11 DBMS and ORACLE on VAX, IDMS (Cullinet), TOTAL, and IMS on IBM]

SC1.8 - Test Bed Terminals
The Test Bed terminals are ASCII terminals which are supported by the vendor supplied host operating systems and which provide the following capabilities:

1. Scroll mode

2. Cursor position control

3. Clear screen

4. Forms mode

In addition, graphic capabilities compatible to the Tektronix 4014 or 4112 are desirable to support the animation capabilities of IDSS\2.0.

SC1.9 - Technology Transfer by 31 March 88
The Test Bed design is constrained to provide integration technology.

SC1.10 - Test Bed Hardware is Expansible
The Test Bed hardware design allows for expansion, and is not restricted to the Early Implementation hardware configuration.

TEST

SC1.11 - Test Configuration
Test programs or selected functions taken from new or existing Application Subsystems are used to test and demonstrate the Test Bed software and concepts.

SC1.12 - Modular Testing, Phased Integration
The design of the Test Bed supports the testing of individual modules and a phased integration.

SC1.13 - Demonstration of Class II Integration
The Test Bed scenarios support the demonstration of Class II integration.

SC1.14 - MCMM, IDSS 2.0 and MRP Demonstration
The Test Bed demonstrates the operations of selected functions belonging to MCMM and IDSS 2.0, in an integrated environment. The Test Bed also includes

specific functions from an MRP system to be selected. [IDSS 2.0 runs as a stand-alone application subsystem]

SC1.15 - MRP Selection Criterion
The MRP system to be selected is subject to the following constraints:

1. Runnable on IBM 3033, under MVS.

2. Relies on CODASYL database manager.

3. Adheres to 6201 Standard Guidelines and Procedures.

DESIGN

SC1.16 - The System and Common Data Control Promotes Integration Through Sharable databases

SC1.17 - The Test Bed Architecture is Layered and Modular

SC1.18 - The Test Bed Software Adheres to the Test Bed Standard Guidelines and Procedures

SC1.19 - The Test Bed Software Is Extensible and Flexible
The Test Bed design is extensible and flexible. That is, new technological developments and added functionality can be readily accomodated.

SC1.20 - Physical and Logical Data Structures and Addresses Are Decoupled

SC1.21 - The Test Bed Software Is Portable
The Test Bed is implemented to maximize software portability. Reliance on host specific features must be minimized.

SC1.22 - Binary Data Transfer
The Test Bed design allows the transfer of binary files.

SECTION 3

SYSTEM SPECIFICATION

3.1 Scope

3.1.1 Identification

This specification establishes the prioritized detailed functional requirements for the DAPro Integrated Information Support System (IISS) Test Bed. This system is intended to be a test computing environment providing integrated data management facilities and distributed processing for heterogeneous databases resident on heterogeneous computer systems interconnected via a Local Area Network.

This document has been prepared by Project Priority 6203 of the Air Force's DAPro Program. This project is being conducted by the Control Data Corporation with the participation of Structural Dynamics Research Corporation, supported by various consultants and contributors.

This project is sponsored by the Manufacturing Technology Division of the Air Force Wright Aeronautical Laboratories.

3.1.2 Purpose

The purpose of this document is to decompose the system requirements to the maximum extent possible, to assure validity of the requirements and to promote maximum understanding between DAPro Program Office personnel, Contro Data and the development subcontractors before system design begins.

3.2 References

3.2.1 Applicable Documents

1. Systran, Inc., ICAM Documentation Standards, _ICAM Document IDS 150120000_, 28 Dec 81.
2. A.D. Little, _ICAM Computer Based Information System (CBIS), System Environment Document_, ICAM Document SED 310140000, Sept 81.
3. A.D. Little, _ICAM Computer Based Information System (CBIS), State of the Art Document_, ICAM Document SAD 310140000, Sept 81.
4. A.D. Little, _ICAM Computer Based Information System (CBIS), System Requirement Document_, ICAM Document SRD 310140000, Sept 81.
5. Control Data Corporation, _System Requirements Document_.

6. General Electric Company, Test-Bed System Requirements Document, Revised June 11, 1982.

7. SofTech, ICAM Test-Bed Interim Standards and Procedures, ICAM Document ISP 620150000, February 1982.

8. ICAM Program Office, The Integrated Sheet Metal Center, 30 Sept 81.

9. N. Tupper, Memorandum for the Record, 5 Oct 81.

10. SofTech, IISS Test Bed Network Transaction Manager System Design Specification, Document 1098-7, June 1982.

3.2.2 Terms and Abbreviations

See Section 1.4 of this document.

3.3 Requirements

This subsection is written from the point of view of the System Engineer for the purpose of allocating functional specifications to the Requirements, Constraints and System Constraints listed in the System Requirement Document (Section 2 of this document). The System Specification is the first document in the life cycle which addresses the implementation aspects of the system design.

In the following, the requirements are identified by unique R numbers (Ex: R3), the constraints are identified by unique C numbers (Ex C 1-1) and the functional specifications are identified within each constraint by sequential F numbers (Ex: F3).

3.3.1 Functional Requirements

R1 Data Shareability Between New or Existing ICAM Application Processes

C1.0 Query Capabilities

F1 Predefined query capabilities are provided to enable Integrated Processes to query data contained in the Test Bed databases

F2 Predefined query capabilities provided are non-procedural

C1.01 Deadlock Resolution

F1 Prevent system deadlocks by designing out database managers deadlock (Future)

F2 Local host computer system capabilities are relied upon for preventing, detection and resolution of local dead locks

C1.1 Heterogeneous Computer Systems

F1 Provide system capabilities to support conversion of messages to specific host internal representation

C1.2 Heterogeneous Database Managers

No functional requirements identified

C1.3 CODASYL Databases / Database Structures

F1 The following CODASYL databases are supported:

VAX/VMS	IDBMS	FOR	IDSS III.2.0
	VAX11 DBMS	FOR	MCMM
IBM 3033	IDMS	FOR	MRP System
LEVEL 6	IDS II	FOR	MCMM

F2 The design will be extensible to allow the following non-Codasyl database management systems to be supported:

IBM 3033	IMS
IBM, VAX, UNIVAC	ISAM FILES

[Release 2.0: The design was extended to support ORACLE on VAX, and TOTAL and IMS on IBM]

C1.4 Traceable Messages

F1 Messages are uniquely identified
F2 Messages are tagged by origin
F3 Messages are tagged by destination
F4 Message logs are selectively accessible
F5 Message logs can be sorted in chronological order
F6 Message logs can be sorted by origin
F7 Message logs can be sorted by destination
F8 Message logs are selectively cleared
F9 Message logs are centrally accessible (Future)
F10 Message logs can be enabled/disabled by the Common Data Administrator (Future)
F11 Messages are defined in the Common Data Model (Future)

C1.5 Query Transactions

F1 Predefined retrieve only application processes will be supported by end of contract
F2 Query application processes to Class II data are specified in the Neutral Data Manipulation Language (NDML)
F3 The Neutral Data Manipulation Language (NDML) is non-procedural (non-navigational)
F4 Query format is independent of target database

F5 Query implementation is independent of target data structure
F6 Support migration to ad-hoc query
F7 Access data contained in more than one database
F8 Support select capabilities
F9 Support join capabilities
F10 Support project capabilities
F11 Perform unit conversion between data units (example metric/ English). The data required to support the data conversion operations is defined in the CDM
F12 Error status can be provided to query originator
F13 The data checking operations defined in C1.7
F14 The common data queried by the query application processes is described in the Common Data Model

C1.5.1 Integrated Application Processes

F1 Integrated applications use query transactions (C1.5) to query data contained in Test Bed databases.
F2 A pre-compiler is used to preprocess new applications which utilize neutral DML statements.
F3 By end of contract, the pre-compilation may be manually performed. [Extended to be automatic]
F4 Integrated application processes query Class II data.

C1.5.2 Non-Integrated Application Processes

F1 Non-integrated application process queries and updates the database to which it is bound.
F2 Non-integrated application processes are not supervised by the Common Data Model processor.
F3 Data integrity checks may be performed on data supplied to a non-integrated application process.
F4 Data integrity checks are performed on data supplied by a non-integrated application process to the Test Bed. (Future)

C1.6 Migration to Class I Data

F1 The Test Bed design supports migration to Class I data environment.

C1.7 Class II Data Integrity

F1 Class II data integrity checks include:
o Edit Checking (Ex: 7 Alpha Characters)
o Domain Checking (Ex: shirtsize: small, medium)
o Range Checking (Ex: o, n, 15)
F2 Data integrity checks are performed selectively. The information required to support and control these checks are contained in the CDM. This

information may be distributed to improve performance. Data checking selection is made by query.

F3 Standard error messages are issued to the data query originator when data integrity checks are violated. The text of the error message is contained in the CDM.

F4 Error correction/disposition action is selected and performed by the data query originator

F5 Error messages are logged and are selectively accessible from a central location. [Release 2.0 - Accessible on each host]

C1.7.1 Message Integrity Checking

F1 Message integrity checks include:
- o Edit Checking
- o Domain Checking
- o Range Checking

F2 The format of the messages and information required to support and control message integrity checking is contained in the CDM.

F3 Standard error messages are issued when message integrity checks are violated. When possible, the message originator is notified of the violation.

F4 Error correction/disposition action is selected by the system administrator, and is implemented by the distributed services of the Test Bed.

F5 Error messages are logged and are centrally accessible. [Rel 2.0 - Accessible on each host]

F6 Message integrity checks can be enabled/disabled by message type, by source and by destination. (Future)

C1.8 Minimize Modifications to Host Operating Systems

F1 Modifications to the host operating systems will be limited to I/O device drivers, and these modifications will be made only if they are necessary.

C1.9 Minimize Modifications to Existing Application Subsystems

F1 Redirection of application processes terminal input/output on the host computer systems may be required.

C1.10 Transparent Data Location

F1 NDML query format is independent of data location

F2 Data locations are contained in the Common Data Model

F3 Integrated application processes need not know the location of Class II data

C1.11 Transaction Processing

F1 Application processes perform predefined data processing tasks.
F2 Only predefined application processes are provided by end of contract.
F3 Application Processes can execute without user interaction.

C1.12 Prioritized Transaction Processing

F1 Application process release to the local host operating system can be scheduled by wall clock time, or by relative priority within the host. (Future)
F2 Application process execution is controlled by the host, in its local environment.
F3 Application process priority is assigned by the CDM Administrator and is stored in the CDM. (Future)
F4 Application process priority information can be defined by message type.

C1.14 Active Data Security

F1 Active data security is provided. (Future)
F2 For predefined application processes, access to common data is granted via the schema and application process logic.
F3 Access to predefined application processes is controlled on a user role basis.
F4 Data supporting application process access control is contained in the Common Data Model, and can be changed by the CDM Administrator. (Future)

R2 Provide Common User Interface to New or Existing Application Processes

C2.0 Common Terminal Interface

F1 A common terminal is used for input and for output to Test Bed functions, integrated and non-integrated application processes.
F2 The common terminal supports the following functions: clear screen, cursor positioning, scroll mode, and is supported by VAX VMS.

C2.1 Virtual Terminal Interface

F1 The virtual terminal defines a standard set of terminal features supported by IISS terminals and recognized by IISS application programs.
F2 The virtual terminal implements standard communication protocol to communicate between itself and the hardware terminal. ANSI terminals

such as the VT-100, VT220, VIP7200 and ADM-3 are supported.

F3 The virtual terminal defines a standard communication protocol to communicate between two virtual terminals.

F4 The virtual terminal defines a set of control words.

F5 The virtual terminal maps specific hardware terminal control characters into the corresponding virtual terminal control words.

F6 The virtual terminal maps the virtual terminal control words into specific hardware terminal control characters.

F7 The virtual terminal implements the bidirectional virtual terminal communication protocol (defined in F3).

F8 The virtual terminal implements the bidirectional virtual to hardware terminal protocol supported by the specific hardware terminal (defined in F2).

F9 The conversions of virtual terminal control words into control character conversions are table driven. (Revised to program code)

F10 The implementation of the virtual terminal to hardware terminal protocol is table driven. (Revised to program code)

F11 The virtual terminal defines and implements a standard data structure representing the terminal image.

F12 The Common Data Model contains administrative utilities for maintaining information relative to terminal types. (Future)

F13 The information relative to terminal types etc. is provided to the virtual terminal by the Common Data Model. (Future)

C2.2 Consistent Invocation and Control of Application Subsystem

F1 The User Interface provides the ability to invoke all application processes in a consistent fashion.

F2 The User Interface is independent of the host on which the application process of interest resides.

F3 The user need not know on which host the application process of interest resides.

C2.3 Invocation and Control of Test Bed Utilities

F1 The User Interface provides the ability to invoke all Test Bed Utilities in a consistent fashion.

F2 The User Interface is independent of the host on which the Test Bed Utility of interest resides.

F3 The user needs not know on which host the Test Bed Utility of interest resides.

F4 The Test Bed utilities can be invoked in a novice or in an expert mode.

F5 The User Interface validates the entries made by the user. Error messages are issued when appropriate and are contained in tables maintained by the Common Data Model. [Message Management tables independently managed]

F6 The User Interface is tutorial through a help facility.

C2.4 Extensibility, Flexibility of the User Interface

F1 The User Interface provides a help facility. The help facility makes use of help messages contained in the Common Data Model.

F2 The User Interface allows the definition of new application processes, and new Test Bed Utilities.

F3 The User Interface services may be invoked by an Integrated Application for user interaction.

F4 The semantic contents of user entries recognized by the User Interface must be readily changeable.

F5 User entry syntax and semantics to invoke application processes and Test Bed services are stored in the Common Data Model. [Defined in UI and forms]

F6 The textual contents of the User Interface forms is supplied to the User Interface by the Common Data Model. [Defined in forms definitions]

F7 A form restore/refresh capability is provided.

F8 The Help Utility can be invoked within a sequence of forms.

F9 The User Interface provides a Help Facility which is table driven. The Help Facility can be invoked without exiting the command mode of the User Interface. Exiting from the Help Facility leads back to the command mode (Indirection).

C2.5 Application Subsystem Security

F1 Access to the various application processes is controlled by user role. User role is established by the user logon process.

F2 The information required to support the Test Bed application process access control is contained in the Common Data Model. [In UI files]

F3 The Common Data Model contains the administrative utilities to update, delete and modify the access control data (Future)

F4 Access to application processes are logged. The log contains sufficient information to uniquely identify the user.

F5 Access to the Common Data Model administrative utilities is controlled. User access authorization is required.

C2.6 Centralized User Interface Support Data

F1 The data required to support the User Interface is contained in the Common Data Model. (Future)
F2 The Common Data Model processor contains the administrative utilities to update, delete, and modify the User Interface support data (security, syntax, semantic contents, etc.).
F3 Access to the administrative utilities used to modify the User Interface data is selective.

C2.7 Forms Generator

F1 The forms generator is used to define, modify, delete and test forms.
F2 The data used to define a form is stored in the Common Data Model. (Future)
F3 Access to the forms generator administrative utilities is selective.
F4 The forms generator allows the definition of a form by using forms already defined.
F5 A form is uniquely identified by its name.
F6 A form consists of fields, each of which is characterized by its attributes.
F7 The forms generator allows the definition of attributes to be decided and selected from the following list:

o Field Attribute	o Definition
Protected/unmodifiable	Cannot be changed
Numeric	Contains only numerals
Alpha	Contains only alphabetic characters
Non-displayable	Used for entry passwords
Non-printable	Cannot be printed locally
Right justification	Character string may be right justified
Left justification	Character string may be left justified
Mandatory entry	Entries must be made before transmission
Selectable	Light pen or equivalent can be used (Future)
Not Transmittable	For local display only
Blinking/reverse video	Contents of field are highlighted
Scroll	Field is overwritten as a scroll
Page	Field is overwritten as a page

Top justifiable	Lines may be justified to top of field [NA - Delete]
Bottom justifiable	Lines may be justified to bottom of field [NA-Delete]
Color	Color of characters when displayed
Intensity	Brightness of field when displayed
Background	Color of background of field
Variable description	Description of the variable contained in the Common Data Model

F8 The forms generator defines forms in a manner that is not specific to a given hardware terminal. The definition is made in terms of the control words recognized by the Virtual Terminal Interface.

F9 The forms generator allows the display of a form on the terminal of an interactive IISS user.

F10 Field information defined in forms that are currently active is available to the application.

F11 The application process which invoked the User Interface can obtain the name of the form currently active.

F12 The forms generator performs data integrity checking on one or more fields defined within a previously displayed form.

F13 The forms generator writes information to one or more fields defined in the current and in the previous active form (i.e., two forms may be accessed).

F14 A list of nested forms can be displayed. Facility is provided to select the form to be returned to.

F15 The forms generator can be invoked by an application subsystem or a Test Bed utility. The name of the form to be specified is supplied by the user.

F16 The forms generator provide diagnostic messages to assist in the definition of forms. Overlapping and adjacent fields are detected.

F17 The forms generator allows the definition of the size of the largest form to be displayed. Error messages flag overcrowding violation.

F18 The forms generator supports the following run-time diagnostics:
- o Form not supported
- o Form not displayed

F19 The forms generator allows for the testing of forms.

C2.8 Report Generator
Not included in the scope of the current contract.
[Was included in Release 2.0]

R3 Measure and Report System Performance

C3.1 Measure Elapsed Time and Response Time For Processing a Transaction Within the Time Resolution of the Host Operating Systems (Future)

F1 Elapsed times for processing an application process are measured within the time resolution of the host operating system.

C3.2 Measure Transaction Frequency, Elapsed Time, Response Time, and Provide Selective Reporting (Future)

F1 Provide application process frequency, elapsed time and response time statistics by application process source and destination.
F2 Provide totals, means, standard deviations of application process measurements by source and by destination.
F3 Application process statistics are provided over the period of interest.
F4 Application process time measurement resolution: TBD.
F5 Application process classification includes: system and user classes. Additional subclasses of application processes for performance monitoring purposes: TBD.
F6 Performance measurements are selectively enabled, at the system level, by the system administrator.

C3.3 Measure Test Bed Resource Usage, and Provide Selective Reporting

F1 Measure Test Bed resource (hardware, software) usage selectively, by application process. (Future except for tests with special performance tools)
F2 Test Bed resource measurements include Test Bed hosts, and Local Area Network as provided by the Vendors.

C3.4 Use Vendor Supported Resources Monitors

F1 The execution of vendor supported software resource monitors can be controlled from Test Bed host terminals.

F2 Resource monitoring statistics can be obtained from Test Bed host terminals.

C3.5 Support User Accountability

F1 Provide audit trail linking users to data entered. (Future)
F2 Provide log of application process usage by users and by time of day. (Partial - information in messages logs)

R4 Provide Test Bed System Control

C4.1 Provide Communication Between Application Processes

F1 Application processes are uniquely addressable.
F2 Application process addresses can be used to identify the host running the application process.
F3 A Local Area Network is used to communicate between hosts. (Partial)
F4 The modes of host to host communications (virtual circuit, datagrams) supported by the Test Bed are TBD.
F5 Error messages are logged. Error messages identify the requesting process.
F6 Messages are classified into classes and subclasses to be decided. Message processing functions are settable by class, subclass, origin and destination TBD.
F7 Messages are prioritized, according to scheduled time of execution (within seconds), relative priority levels (256/levels), or else are processed on a first-in, first-out basis. (Partial - uses FIFO only)
F8 Messages are classified into classes, subclasses, to be decided.
F9 Messages are prioritized. (Partial)
F10 Asynchronous communication environment is provided.
F11 Application process Priority Levels and priority schemes are TBD. (Future)
F12 Message queues are recoverable. (Future except Guaranteed Delivery messages)
F13 Messages are selectively acknowledged to the source. Message acknowledgement scheme is TBD.
F14 Transmission error checking is provided. Error checking methods are TBD.
F15 Error messages are stored in the CDM. (Future)
F16 The guaranteed delivery of messages is provided selectively. The information controlling the guaranteed delivery service is contained in the CDM (Future). Guaranteed delivery control information can be defined by message type, source, destination, and is carried in the message. (Message type only)

C4.2 Provide Active Routing of Transaction Messages

- F1 The automatic routing of the messages is provided.
- F2 The information required to route the messages is contained in the Common Data Model. [NTM System Data, not CDM]
- F3 Routing of a message to a non-available or non-existing resource results in an error message being sent to the originator. (Partial)

C4.3 Control Execution of Application Processes

- F1 Messages are used to load, initiate, cancel a specific application process.
- F2 Messages are used to control the redirection of the I/O of a specific application process. (Future)
- F3 Messages controlling the execution of application messages are selectively acknowledged by the receiver: status information and error messages are provided to the sender.
- F4 Error messages with unique identification of the sender and receiver processes are logged and are centrally accessible. [Accessible on each host]
- F5 Messages are prioritized. (Partial)
- F6 Abort messages have the highest priority. (Future)

C4.4 Provide Error Processing

- F1 Error detection is provided during the retrieval and manipulation of information contained in the Common Data Model.
- F2 Error messages resulting from transmission errors or Common Data Model errors (processing or data) are logged, and are centrally accessible. If possible, the originating user or process will be notified. The interpretation of error messages and their resolution is left to the originating user or process if it can be notified. Otherwise, the interpretation of error messages is performed by the Test Bed distributed services. [Error Log per host only]
- F3 The Test Bed supports the same level of error processing capabilities than that provided by the application program prior to their installation.
- F4 Error calls to local host operating systems are trapped and handled as described in F2. (Partial)

C4.5 Maintain Test Bed Operability

C4.5.1 Test Bed Cold Start

- F1 Provide IISS bootstrap capabilities on all IISS hosts.

F2 Test Bed cold start functions support:
1. Host to host communication (LAN Start-up)
2. Access to Common Data Model for all hosts
3. User interface
4. Resource and performance logging (Future)
5. Downloading of IISS configuration data (LAN) from CDM (Future)
6. System synchronization (TBD)
7. System coordination (restart, shutdown)

F3 Completion of the cold start phase is detected and reported to the Operator Console.

F4 Abnormal conditions encountered during cold start are logged and displayed on the system operator terminal

F5 Provide LAN bootstrap capabilities

C4.5.2 Test Bed Shutdown

F1 Provide for the orderly, graceful shutdown of the Test Bed

F2 Test Bed shutdown does not result in loss of user data [except abnormal shutdowns]

F3 Test Bed shutdown completion is detected and reported to the Operator's Console

F4 Test Bed shutdown is initiated by the Test Bed operator

F5 Control access to the Test Bed shutdown function

F6 Shutdown of the Test Bed occurs after the system reaches a quiet point [within time limit]

F7 Initiation of the Test Bed shutdown sequence inhibits further Test Bed user logins

F8 Test Bed users are notified when a shutdown sequence is initiated and given reasonable time to finish up

C4.5.3 Test Bed Host Restart

F1 Provide for the restart of IISS services on a host by host basis

F2 Provide for cold start of one host while the Test Bed is running on other hosts

F3 Report completion of cold start of isolated host to the User Interface and to the Test Bed operating system [System operator only]

F4 Test Bed host restart can be initiated from:
a. The host terminals (privileged function)
b. The IISS terminals (privileged functions) (Future)

F5 Reset status information

C4.5.4 Test Bed Selective Host Drop Off

F1 Provide for termination of Test Bed services on a host by host basis

F2 Test Bed shutdown is initiated either from:
a. The host terminals (privileged function)

b. The Test Bed terminals (privileged function) (Future)

F3 Selective host drop off is orderly and graceful, without loss of user data [except abnormal shutdowns]

F4 The selective shutdown of a host is reported to the User Interface and to the Test Bed system [operator only]

C4.5.5 Test Bed Back-Up/Restore

F1 Back-up facilities are provided for:
1. The Common Data Model
2. The Test Bed databases
3. The user application process files
4. The Test Bed service files (statistics, audit trail, etc.)

F2 Back-up can be initiated from the user interface (Future)

F3 The Test Bed back-up facilities use existing host and data management system back-up facilities

F4 Restore capabilities are provided for the files listed in F1

F5 Restore capabilities use the vendor supplied restore capabilities

C4.5.6 Test Bed Coordination and Control

F1 Initiate application process or Test Bed utility
1. Load
2. Execute
3. Redirect I/O
4. Abort

F2 Receive and route messages

F3 Generate and distribute status information

F4 Receive and interpret status information

F5 Queue messages and regulate flow

F6 Trap host operating system error messages and generate status information (Partial)

F7 Guarantee delivery of messages

F8 Provide & record application processes completion status

F9 Monitor resource status. Obtain and provide status information

C4.5.7 Perform IISS Recovery Operations (Future)

F1 Support manual and automatic initiation of recovery operations.
Recovery scenarios: TBD. (Partial - Manual)

F2 Support system wide synchronization of application process logs, databases and journals

F3 Report completion and status of recovery operations

F4 Coordinate with recovery facilities provided by local host operating systems and by local database management systems

F5 Support manual selection of recovery synchronization point
F6 Provide system wide sequencing of pending messages
F7 Acknowledge system wide synchronization points
F8 Execute system wide synchronization point
F9 Involke and terminate IISS systemwide Quiet Points. Quiet Points may be controlled by the Test Bed Administrator/Operator or by a Test Bed program.
F10 Define other recovery functions to be supported by the system.

R5 Maintain Test Bed System Control Information

C5.1 Centralized System and Common Data Control
F1 The information required to control system and common data is centrally maintained and controlled (Partial - System data is distributed)
F2 Access to the Common Data Model is controlled
F3 Information contained in the Common data Model may be distributed to improve performance (Future)
F4 Versions of the Common Data Model are supported (Future)
F5 The Common Data Model can be backed-up and restored

C5.2 Controlled Access of Systems and Common Data Control Information
F1 The description of Common Data is protected and is supplied to authorized users only
F2 Common Data Model processor supports asynchronous processing environment
F3 The Common Data Model supports error processing
F4 The Common Data Model supports multiple user processing
F5 The Common Data Model acknowledges all requests, either with data or status information

C5.3 Flexibility of the Common Data Control Structure

F1 The Common Data Model can describe a wide variety of data structures
F2 The Common Data Model can be expanded without disrupting Test Bed services

C5.4 Test Bed System Control Migration

F1 The structure of the Test Bed system control information allows migration to a back end database machine

C5.5 The Maintenance of the Test Bed Control Information is Computer Assisted

F1 Interactive maintenance of the Common Data Model and System Control Information
F2 Computer assisted maintenance of the Common Data Model and data includes: creation, deletion, updating, editing
F3 Provide mechanisms to verify the validity of the Common Data Model. The mechanisms to be provided must allow mechanization of their implementation
F4 Provide listings of the Common Data Model
F5 The effectivity of update versions of the Common Data Model is controlled by the Common Data Administrator

C5.6 A Common Data Model Supports the System and Common Data Control Information

F1 The Common Data Model defines the entity classes and relations among them
F2 Mappings between entity classes and their attribute classes
F3 Constraints defining permissible values for attributes
F4 The Common Data Model contains the syntax and semantics of the user form entries (Future)
F5 The Common Data Model contains help data
F6 The Common Data Model contains pre-defined forms (Future)
F7 The Common Data Model contains a description of the Test Bed Network (Future)
F8 The Common Data Model contains the virtual terminal data (Future)

R6 Provide Capability to Implement and Test Programs on the Test Bed

C6.1 Protect System from New Application Programs

F1 During testing, application programs are prevented from changing system data (Future)

C6.2 Provide Real Data for Testing

F1 Application programs under test may use real data for input
F2 Application programs under test are prevented from updating the Test Bed databases (Future)
F3 Application programs under test direct their output to a test database or terminal (Future)
F4 Application programs are declared in test mode via the Common Data Model by its Administrator (Future)

C6.3 Make Maximum Use of Vendor Supplied Utilities

F1 Maximum use is made of Vendor supplied utilities for process implementation and testing

C6.4 Allow Analysis of Message Journals

F1 Message journals can be sorted by source, destination and by message type
F2 Sorted message journals can be output to a terminal selected by the user
F3 Sorted message journals are cleared at the option of the user

C6.5 Test Bed May Be Shutdown for Special Conditions

No functional requirements identified

C6.6 Test Results Shall be Accessible (See C6.2 - F3)

C6.7 Tests Shall be Repeatable

F1 Copies of system databases can be created to provide stable test data for test purposes
F2 The system database copies are created at the request of the user (Partial)
F3 The system database copies are released at the request of the user (Partial)
F4 The system database copies can be write protected (Partial)
F5 The version of the Common Data Model being used is accessible to the user (Future)

C6.8 Test Protection Modes Shall be Easily Changeable

F1 The test protection modes are settable via the CDM
F2 The test protection modes include: write protection of output, redirection of output to a test output file, or to the test input file (in the case of an input/output file)
F3 Test protection modes may be declared without recompiling the programs under test [Test mode requires program recompilation]

3.3.2 <u>Physical/Performance Requirements</u>

3.3.2.1 <u>Physical Requirements</u>

Project 6203 Test Bed requirements are two host computers interconnected via a Local Area Network:

o VAX 11/780
o IBM 3033 (or 30xx)

The Test Bed system and application processes must be controlled from a single terminal. The early implementation relies on VT-100 terminals for demonstration purposes. In keeping with the concept of the Virtual Terminal described in this document, none of the application processes or Test Bed system utilities is relying on the non-standard (as defined in the Test Bed) hardware features provided by the VT-100.

3.3.2.2 Performance Requirements

The Test Bed is experimental in nature, and its expected level of performance cannot be quantified. Careful design trade-offs are made to minimize user responses and elapsed time.

To allow optimization and experimentation, a number of Test Bed processing features (message logging, for example) are settable at the system level via the Common Data Model. (Future)

Furthermore, the Test Bed design allows for improving performance through reconfiguration of the hardware and software. The portability of the application processes facilitate reconfiguration.

3.3.3 Interfaces : Requirements

3.3.3.1 Test Bed User Interface

The user interfaces with the Test Bed via the User Interface. The functional requirements of the User Interface have been described in Subsection R2, "Provide Common User Interface to New or Existing application processes". The user interfaces with the Test Bed via a terminal. The early implementation Test Bed accommodates DEC VT-100 terminals. There are no additional hardware interface requirements. The data supplied by the user are either system commands or application process commands and data. The nature and syntax of the system commands are to be decided. On the other hand, the nature and syntax of the existing application process commands and data is already defined and will not be modified.

3.3.3.2 Common Data Model Administrator

The Common Data Model Administrator is entrusted with the definition and population of the Test Bed Common Data Model. The Administrator interfaces with the Test Bed Common Data through a terminal, and exercises administrative utilities which have been described functionally in Subsection 3.3.1. The method of interaction of the CDM Administrator and the CDM is TBD. It is anticipated that the CDM Administrator performs the majority of his/her work interactively.

3.3.3.3 Application Subsystem Interfaces

Specific interface prog ıms are used to interface selected application processes to the Test Bed. These programs are using data provided by the Common Data Model to control the data brought into the Test Bed. From a hardware point of view, the interfacing is implemented via magnetic tapes, disk drives, or

ASCII communication lines. The exact format, encoding of the common data required by the interface programs is entirely dependent upon the application processes which are interfaced. Hence, the Test Bed must support the transfer of binary information.

3.3.4 Technology Voids

In the context of this system specifications, technology voids (TV's) are those activities which present significant challenges from a technology or implementation point of view. These activities are listed here because of their impact on the completion of the Test Bed project.

3.3.4.1 Communication Technology

3.3.4.1.1 Host-to-Host Communication

TV: Define and implement a set of communication services which satisfy the following requirements:

1. Provide reliable host to host communication services.

2. Are implementable, with minimum modifications, on the VAX 11/780, the IBM 3033, the Honeywell Level 6.

3. Are implemented with minimum of modification to the Host Operating System and terminal drivers.

4. Are implementable in a High Order language (FORTRAN, COBOL) whenever feasible. Assembly language routines may be required.

5. Support communication protocol (virtual circuits or datagrams) to be decided.

6. Are upgradable to high speed data transfers.

7. Are modularized to contain hardware dependencies.

8. Are controllable and manageable.

3.3.4.1.2 Guaranteed Delivery

TV1: Define and implement procedures and algorithms required to guarantee the delivery of messages within the Test Bed. Message delivery must be guaranteed even if the destination process is temporarily unavailable.

3.3.4.1.3 Process to Process Communication

TV1: Define and implement a set of communication services which satisfy the following requirements:

1. Provide process to process communication services

2. Are implementable, with minimum modifications, on the VAX 11/780, the IBM 3033, the Honeywell Level 6

3. Are implementable on the various Host Operating Systems without modifications to the host OS (other than I/O drivers)

4. Are callable from FORTRAN and Cobol programs

3.3.4.2 <u>IISS System Control</u>

3.3.4.2.1 <u>System Control Strategy</u>

TV1: Define and implement a System Control Strategy which provides:

1. Message and application process control

2. Deadlock prevention

3. System quiet point

4. Coordination of commands, status and responses

5. Hardware resource status acquisition

6. Guaranteed delivery of messages

3.3.4.2.2 <u>System Restart/Recovery</u>

TV1: Define and implement the procedures required to provide a coordinated and synchronized restart of the Test Bed. Procedures support:

1. Execution of synchronization point

2. Roll back to synchronization point

3. Orderly restart of Test Bed services

4. Synchronization of application process logs

5. System status broadcasting

3.3.4.2.3 <u>Databases Recovery</u>

TV1: Define and implement the procedures required to support the recovery/synchronization of the Test Bed data. Procedures support:

1. Synchronization of databases with application process logs and application process queues on the same node.

2. Synchronization of databases application process logs and application process queues at the system level.

3.3.4.3 Schema Definition

3.3.4.3.1 Conceptual Schema Definition and Storage

TV1: Define the techniques and procedures to be used to create an integrated conceptual schema from existing application schemas.

TV2: Define and implement procedures to support the definition effort of a conceptual schema.

TV3: Define a methodology to show a conceptual schema in a machine useable form. This definition of the conceptual schema supports the description of the entities, relation among entities and their attributes, range of allowable values for the attributes.

This methodology is sufficiently flexible to describe the data structures which can be supported on CODASYL database managers, and allows for the description of data with synonyms, homonyms, scale differences, structural differences, and abstraction differences.

3.3.4.3.2 Define a Distributed Schema Architecture

TV1: Define a schema architecture that extends the benefits of the CODASYL three schema architecture to the distributed environment.

3.3.4.3.3 Define Schema Mapping Transformations

TV1: Define the transformations required to support the schema architecture defined in 3.3.4.3.2.

To be practical, these transformations must be computer implementable, and be non-ambiguous.

Ideally, the transformations are performed from data provided by:

1. The various schemas defined in 3.3.4.3.2.

2. The conceptual schema defined in 3.3.4.3.1.

These transformations take into account homonyms, synonyms, and various machine representation of the data.

3.3.4.4 Neutral Data Manipulation Language

TV1: Define and implement system wide data manipulation primitives allowing the definition of set oriented data manipulation operations which satisfy the following requirements:

1. Non-navigational

2. Extendable to the ad-hoc query environment

3. Selection, join, project capabilities

4. Programmer friendly non-procedural

5. Predefined error message specifications

3.3.4.5 Data Retrieval Operation

3.3.4.5.1 Mapping of Data From Single Off-Node database

TV1: Define and implement the data mapping algorithms required to transmit data to a requesting program from a single, off-node database. The mapping algorithms must take into account the differences in schemas, and data characteristics which may exist between the database and the application program requesting the data.

3.3.4.5.2 Mapping of Data From Multiple Off-Node databases

TV1: Define and implement the data mapping algorithms required to transmit data to a requesting program from multiple off-node databases. The mapping algorithms must take into account the differences in schemas, data characteristics which may exist between the databases, and the application program requesting the data.

TV2: Define and implement the procedures and algorithms required to aggregate the data provided by multiple off-node databases into a single data packet useable by the requesting program. Support adequate level of error processing to detect and flag possible errors occurring in the various databases. Status information will be provided to the requesting application process.

3.3.4.5.3 Data Constraint Checking

TV1: Define and implement the strategy used to provide data constraint checking. This strategy reflects concerns for:

1. Overall data integrity

2. Ease of implementation

3. Overall system throughput

The data required to support the data constraint checking is supplied by the Common Data Model.

TV2: Data Constraint checking is selectable. The mechanisms responsible for the selection and implementation of the data constraint checks must be identified.

3.3.4.6 Automatic Translation of Data Manipulation Statements

TV1: Define and demonstrate the algorithms and procedures required to support the automatic translation of data manipulation statements into procedures targeted toward a single CODASYL database manager. The translation algorithms must be practical and capable of accommodating the data structures defined by the various schemas. Adequate level of error processing must be provided.

TV2: Define and demonstrate the algorithms and procedures required to support the automatic translation of data manipulation statement into procedures targeted toward multiple CODASYL database managers. The translation algorithms must be practical, capable of accommodating the data structure, defined by the various schemas, and provide the information required to control the aggregation of the data retrieval operations. An adequate level of error processing must be provided.

3.3.4.7 User Interface Command Form Processor

TV1: Define and implement a command form processor sufficiently flexible to allow for the definition of command menus and command semantics in tables.

3.3.5 Data Requirements

3.3.5.1 Logical Organization of Static System Data

The static system data is the data used for reference during operations. It characterizes system parameters and attributes.

The following classes of system parameters, attributes and information included in the static system data:

1. Schema definitions
2. Message definition
3. Screen data
4. User description
5. Network description
6. Virtual terminal data
7. User command language syntax and semantics

The static system data used to describe the Test Bed is contained and defined in the Common Data Model. This data is declared and updated by the Common Data Model Administrator.

The static system data may be distributed by the Common Data Model Processor to improve performance or to reduce software complexity.

3.3.5.1.1 Schema Definition

The Test Bed uses a three schema architecture to facilitate integration. The schema definition data allows the definition of the schemas and of the implied relationship existing between the various schema types existing in the Test Bed.

The following data is required to define the schemas of the Test Bed:

- o Entity class description (Neutral Conceptual)
- o Attribute class
- o Entity/attribute relationship
- o Entity/entity relationships
- o Attribute domain description
- o Key, inherited key classes
- o Entity class description (Neutral External)
- o External/conceptual attributes relationships
- o External schema access authorization
- o Conceptual/internal schema representation
- o Entity class description (Internal)
- o Pathing assertions

The above listed data must be obtained to support the Test Bed demonstration scenarios.

3.3.5.1.2 Message Definition

Test Bed message headers contain information that reflects the characteristics of the Test Bed. This information is obtained from the Common Data Model. [System Data is not in CDM]

- o Message routing information
- o Message delivery, acknowledgement, processing requirements
- o Message priority and trigger information (initiator, receiver)

- o Message type
- o Message data access requirements

3.3.5.1.3 <u>Screen Data Definition</u>

The Test Bed user screens are defined at the system level. The following information is required:

- o Screen name
- o Screen security data
- o Screen constant parameters (text, etc.)
- o Screen data items are derived from the conceptual schema

3.3.5.1.4 <u>Test Bed User Definition</u>

The following information is used to define the Test Bed user:

- o User name, password
- o Legal user roles
- o User role privileges access to Application and System utilities

3.3.5.1.5 <u>Network Description</u>

The Network description data defines the configuration of the Local Area Network, the location of System Services (CDM), and of the application processes.

The following data is required to describe the Test Bed Network:

- o Local Area Network configuration data (terminals, hardware, protocol)
- o Host on which Test Bed System Services resides
- o Host on which Test Bed Application Subsystems
- o Host system description

3.3.5.1.6 <u>Virtual Terminal Description</u>

The Virtual Terminal interface requires a description of the real terminal or application program with which it interfaces. This description is provided by the Common Data Model and defines:

- o The features supported by the real terminal

- o The character set of the real terminal
- o The communication protocol

3.3.5.1.7 User Command Menus and Semantics

The syntax and semantics of the user command menus is defined in the Common Data Model. The following information is required: [Future - System Data is not in CDM]

- o User command menus
- o User command semantics
- o User command security data
- o User Interface help messages

3.3.5.2 Logical Organization of Dynamic Input Data

The Test Bed dynamic input data falls into three broad classes:

1. User Test Bed System Commands
2. User application process commands
3. User application process data

For existing subsystems, the structure, syntax of the subsystem commands and data is already in place, and is outside of the scope of the Test Bed.

For new application subsystems, the structure of the subsystem commands will be identical to the organization of the Test Bed commands, and share the same implementation and syntax.

The organization of new application subsystems input data is not defined at this time. It must conform to the Test Bed standards.

The organization and syntax of the Test Bed commands is to be decided. It is known that the Test Bed commands will provide an expert and novice mode of operation, and will interface with a system wide Help Facility.

It is desirable that system commands be entered either from a file or interactively from a terminal.

3.3.5.3 Logical Organization of Dynamic Output Data

The dynamic output data which a user may either receive or select while interacting with the Test Bed falls into four broad classes:

1. Test Bed system service module acknowledgement to user commands

2. Host operating system error/status messages

3. Application process error/status messages

4. Application process output data

As an enhancement, the user may select to direct all four classes of output data to the same device, or may elect to direct the application process output data to a distinct device. This capability prevents the other classes of output data from garbling the output data generated by the application processes. (Partial)

The Test Bed user thus is able to direct output data classes 1 through 3 to either a terminal or to a file of his choice. Likewise, he may direct the output of an application process to a terminal or to a file of his own choice. (Future)

3.3.5.4 Internally Generated Data

The information internally generated by the Test Bed falls into four broad classes:

1. Status information supporting the internal management of the Test Bed

2. Statistics gathered on the usage of resources and response time

3. Error messages (application process and Test Bed Services)

4. Audit trails

The experimental nature of the Test Bed emphasizes the importance of this internally generated data.

This data is stored in separate files, with sufficient supporting information to facilitate interpretation.

The Test Bed error messages are displayed on the user selected device (terminal, file) are simultaneously logged on the Test Bed system error file.

The above classes of internally generated information can be accessed and cleared by authorized users. The information is displayed on a device selected by the user.

3.4 Quality Assurance

The Quality Assurance and Configuration Management activities take place during all phases of the Test Bed development, beginning with Requirement Engineering and ending with Integration. Documentation is viewed as an integral part of the development process rather than an add-on phase to the development cycle.

3.4.1 Requirement Engineering Phase

The requirement engineering phase of a project is important. It must be conducted thoroughly, with method to ensure that the product being specified and design meets the goals of the customer.

The outputs of the Requirement Engineering phase of the Test Bed project are:

1. System Needs Analysis document
2. System Requirements document
3. System Specification document

Generation of the documents: General Electric, Boeing, respectively, were responsible responsible for the production of these documents; for the 6203 project, Control Data, as the Prime Contractor, is responsible for their update and maintenance, other contractors, as listed in the Forword, support Control Data in this effort by supplying specific inputs and documents. A formal review is conducted jointly by Control Data and its subcontractors.

DAPro program office reviews: The above documents are reviewed by the DAPro program office, who may request modifications required to better serve Control Data and its subcontractors shall respond to the request for modifications by amending, when possible, the original documents. This cycle is repeated until the program office is satisfied that the Test Bed, as specified, will meet the Air Force needs.

Configuration Management: Once the above documents have been reviewed and have been accepted by the Air Force, the documents are placed under configuration management control with Systran, the project Librarian. The documents are promptly distributed to any subsequent effort. The documents, once under configuration management control, can no longer be changed casually. Changes must be made through the Change Control function.

Change Control Function: Requests for changes are reviewed by Control Data and its subcontractors for feasibility and impact on cost and schedule. Requests may be formulated by the DAPro program office, Control Data and its subcontractors. Requests for changes that are approved for implementation are documented at the appropriate level and placed under configuration management control. The existing documents and resulting amendments form a new base line on which any subsequent effort is based. The new base line is promptly distributed to all parties.

Review Mechanisms: The review of the documents produced during the Requirement Engineering phase is facilitated by the meticulous and exacting traceability provided in the documents themselves.

Review Procedure: The chief objective of the reviews conducted of the documents produced during the Requirement Engineering phase is to ensure congruence of the Test Bed features and of the Air Force Needs. The review is facilitated by the meticulous and exacting traceability built into the documents. Each function must be traceable to a needs expressed by the Air Force. As the specification and design progress, the review must focus on the soundness of the technical approaches proposed in support of the Air Force needs. Specification validation by inspection is the main procedure used to carry out such reviews.

3.4.2 Design and Implementation Phase

The Test Bed is designed and implemented using the documents produced during the Requirement Engineering phase as a base line. Recall that these documents have been reviewed and validated by inspection and are under Configuration Management control.

The documents produced during the Design and Implementation phase are:

1. System Design Specification
2. System Test Plan
3. Development Specifications
4. Product Specification - as designed
5. Product Specification - as built
6. Unit Test Plan
7. System Test Plan
8. System Test Report
9. Users Manuals

Documents 1 through 4 benefit from the review procedure used in the Requirement Engineering phase.

Item 5 is generated by walking through the programs written to support the implementation of Item 4. These walk throughs are conducted by people other than the programmers responsible for writing the programs. This accepted approach eliminates the well known psychological obstacles which surface when a programmer is asked to criticize his or her own work.

The Product Specification - as built - and the software it describes are jointly placed under configuration control. Modifications to either one requires invoking the change management procedures.

Unit Testing is conducted by the application programmers, since independent testing, although desirable, is a very costly procedure. The Unit Test Plan, is however, generated jointly by the programmer and the Quality Assurance Engineer.

System Testing is conducted under the direction of the System Engineer with support from the Quality Engineer.

3.4.3 Test Bed Software Quality Attributes

The quality of software can be described in terms of quality factors which are mission dependent. Bearing in mind the experimental and changing nature of the Test Bed software, it is required that the Test Bed Software be:

1. Traceable
2. Complete
3. Consistent
4. Accurate
5. Simple
6. Modular
7. General
8. Expandable
9. System software independent
10. Machine independent
11. Terminal independence
12. Use common data definitions
13. Database management system independent

The above criteria will be used when reviewing software as well as when making the unavoidable engineering trade-offs facing designers. Efficiency, although desirable, is not thought to be of sufficient concern to override the above listed quality criteria.

APPENDIX A

TEST BED NEEDS ANALYSIS

A.i Background Information

The Needs Analysis presented in this appendix is founded on the following documents.

1. "The Integrated Sheet Metal Center"
   Source: ICAM Program Office
   Date: 30 September 81
   Document Mnemonic: ISMC

2. "Needs Issues" Documents
   Source: Richard Mayer
   Date: 8-9 March 82
   Document Mnemonic: NEEDS

3. "Memorandum For The Record"
   Source: Nathan Tupper
   Date: 5 October 81
   Document Mnemonic: MEMO

A.ii Needs Analysis Methodology

End User and System Needs definition. The above documents have been reviewed for their needs contents. The explicit and implied needs contained in these documents have been extracted and referenced to form the Test Bed Needs Analysis Document. The Test Bed Needs Analysis Document and the documents prepared for the DAPro (formerly, ICAM) program office, such as the CBIS Needs Analysis, state of the art, and requirement documents, form the basis on which the Test Bed requirement document have been prepared.

A.1 Mission Statement

In the following, needs are listed and explained, then a reference is given to one of the documents listed in section A.i above (i.e., ISMC, Needs, or MEMO) where the need is found.

A.1.1 To Develop and Demonstrate Integration Technology

A.1.1.1 To Systematically Develop Combination of Technologies

Systematically develop combination of technologies required to solve the Information System Integration problem.

(ISMC, III, 17)

A.1.1.2 To Provide Visibility into Cost and Problems of Integration

Provide visibility into the cost and problems associated with doing integration (at least in a test environment).

(Needs, B, ii)

A.1.1.3 To Demonstrate the Flexibility of the Integration Environment

The need to propose and demonstrate advanced information system development concepts (e.g, system development thread) (not how you do it necessarily but how flexible is the resulting environment to change).

(Needs, A)

A.1.1.4 To Demonstrate Advanced Concepts in Information Management

The need to propose and demonstrate advanced concepts in information management (e.g., the Information Management Thread)

(Needs, A, iv)

A.1.1.5 To Demonstrate that Integration is Workable in an Evolutionary Mode

The solution must be workable in current environments. It must also support an evolutionary implementation of the new Information Management Thread (IMT) technology into the existing hardware, software and particularly the management systems of these environments.

(ISMC, III, 17)

A.1.2 To Support the Current Contractor in the Implementation of the ISMC

A.1.2.1 To Support Current and Future Development and Implementation

Establish a center of excellence and a cadre of experienced ICAM companies personnel to support 2201 concept development and 2202 implementation, as well as future projects', such as DAPro, development and implementation.

(Needs, B, i)

A.1.2.2 To Simplify the Task of the Current Project Contractor in Integrating other Integrated Application Subsystems

...Thereby considerably reducing the risk to the successful contractor, ..., and shortening the lead time for implementation of the ISMC.

(Memo, page 1, paragraph 2)

A.1.2.3 To Validate System for Implementation on the ISMC

This is to be a Test Bed for Information Management System concepts, for individual advanced manufacturing systems, and for the initial integration of two or more systems. The intent is that all such systems (or at least most of them) will be run through the Test Bed before implementation in the ISMC.

(Memo, page 1, paragraph 2)

A.1.2.4 To Identify, to Quantify Initial Performance Improvements

Such a Test Bed will identify and solve problems, as well as demonstrate and quantify initial performance improvements.

(Memo, page 1, paragraph 2)

A.1.3 To Support the DAPro Program Activities

A.1.3.1 To Provide IISS Enhancements by April '88

The solution strategy must provide proof of tangible benefits by early\1988 in order to assist in transferring these new enhancements into the ISMC demonstration.

(ISMC, III, 17)

A.1.3.2 To Demonstrate Integration of Manufacturing Applications

The need to demonstrate integration of manufacturing applications.

(Needs, A, ii)

A.1.3.3 To Demonstrate Major Steps in Long Range Paperless Factory

A.1.3.4 To Demonstrate Integrated Application Software Viability

The Need to Demonstrate Integrated Software Viability.

(Needs, A, i)

A.1.3.5 To Run Integrated Systems on Test Bed Before Implementation

A potential user (of Integrated Application Systems) will never be the first user of any integrated product run through the Test Bed even if something (......) is not used in the ISMC, it still will have been integrated into and demonstrated in the Test Bed.

(Memo, page 1, paragraph 2)

A.1.3.6 To Provide a Second Verification and Validation to the Integrated Software

Get a second opinion of Integrated Application developed software systems.(a\backdoor\approach to V&V)

(Needs, B, iii)

A.1.3.7 To Educate the Integrated Application Community to the Total Picture

Educate the Program Office and the Integrated Application community to the total picture.

(Needs, B, iv)

A.1.4 To Support the Following High Level Needs of the Air Force

The Air Force "needs" those needs associated with improving our defense posture.

(Needs, D)

A.1.4.1 To Reduce Weapons Cost

(Needs, D, i)

A.1.4.2 To Increase Surge Capability

(Needs, D, ii)

A.1.4.3 To Improve Planning Capability

(Needs, D, iii)

A.1.4.4 To Improve Capabilities of Defense Contractors

(Needs, D, iv)

A.1.5 To Support the Following High Level Needs of the Aerospace Industry

Manufacturing "needs" those needs associated with improving productivity through the systematic application of SDM and computer technology.

(Needs, C)

A.1.5.1 To Increase Profits

(Needs, C, i)

A.1.5.2 To Improve Responsiveness

To improve responsiveness to change in:

1. Order timing, quantity or configuration
2. Engineering
3. Technology

(Needs, C, ii)

A.1.5.3 To Improve Product Quality

(Needs, C, iii)

A.1.5.4 To Improve Schedule Realization

(Needs, C, iv)

A.1.5.5 To Improve Competitive Position

(Needs, C, v)

NOTE: "The following scenarios are listed here because the Test Bed, per Mission Statement A.1.3 supports the ISMC implementation and development. These scenarios shed additional light on the requirements which must be satisfied by the Test Bed to be of value to ISMC".

A.2 ISMC Scenarios

A.2.1 Product Engineering Needs

A.2.1.1 To Provide for Manufacturing Engineering to Review Designs

To provide for manufacturing engineering to review designs and design changes for producibility prior to release of firm design.

A.2.1.2 To Provide for Coordination of Engineering Changes

To provide for coordination of engineering changes with all affected activities prior to release to determine impacts of engineering changes on schedule, costs...

A.2.1.3 To Establish Effectiveness Based on Type of Change

To establish effectiveness based on type of change, cost of retrofit, versus cost of production installation, availability of new parts and customer requirements.

A.2.1.4 To Withdraw an Old Bill of Material

To withdraw an old bill of material and insert a new bill of material on a net change basis.

(ISMC, III, 5)

A.2.2 Manufacturing Engineering Needs

A.2.2.1 To Provide History of the Activities

To provide a history of the activities which led to the design of the center, including alternatives considered or rejected.

(ISMC, III, 7)

A.2.2.2 To Provide a Description, in Detail

To provide a description, in detail, of the operating strategy (or strategies) for the proposed center, including the strategies to control and handle material and tools in the material handling system, and strategies for machine tool processing and quality control.

(ISMC, III, 7)

A.2.2.3 To Conduct Appropriate Simulations

Appropriate simulations of the proposed center and elements of it which address system configuration, operating strategies, parts spectrum and performance measures. The performance measures to be addressed should include at least the direct and indirect hours of personnel, scrap hours, throughput time, machine utilization, span time, queue time, jobs completion route, number of late/expedited jobs.

(ISMC, III, 7)

A.2.2.4 To Consider Those Automation Priorities Defined in 2103

In addition, the design of ISMC shall consider those automation opportunities identified in ICAM Project Priorities 2103, 2108, 9104 and 9301.

(ISMC, III, 7)

A.2.3 Process Planning Needs

A.2.3.1 To Provide Direct Access to Logical Data

The ISMC must contain an interface to the Process Planning activity of the technical thread to provide direct access and transfer of the logical data generated in Process Planning.

(ISMC, III, 8)

A.2.3.2 To Provide Automation Communication to Create Work Instructions

It is required that the capability exist for automated communication, storage and retrieval of the appropriate information in order to create work instructions for the shop floor.

(ISMC, III, 8)

A.2.3.3 To Structure and to Store Process Planning Data

Process Planning data must be structured and stored to support the operation of numerical control processes by on-line communication of information to numerical controlled processes and control functions.

(ISMC, III, 9)

A.2.4 Master Schedule Planning Needs

.2.4.1 To Simulate the Effects of Proposed Changes on Production Resources

The Master Schedule Planning activity shall have the apability to simulate the effects on production resources, including machine types, personnel, inventory levels, of proposed changes in the production schedule or product mix.

(ISMC, III, 9)

A.2.4.2 To Load and to Execute Simulation in Batch Mode

The simulation can be loaded and executed in a batch mode, provided that on-line access to such information as Schedules, current resources status, ..., can be accomplished.

(ISMC, III, 9)

A.2.4.3 To Receive the Indentured Bill of Material from Manufacturing Planning

Production Requirements Planning from Manufacturing Planning and Process Planning, it receives the manufacturing indentured bill of material, and information regarding set-up time, run time per piece, and material move time.

(ISMC, III, 10)

A.2.4.4 To Retrieve Recent Actual Manufacturing Statistics

Where a significant variance exists, Production Requirements Planning could then examine the factors on which the forecast was made (such as set-up time, run time, scrappage and the like), call up the recent actual data on these factors, and update where necessary the planning factors using recent actual data.

(ISMC, III, 10)

A.2.5 Capacity Planning

A.2.5.1 To Access Automatically the Inventory Requirement Files

To include the following (minimum) capabilities:

1. Automatically access inventory requirements files and process planning files and then convert production requirements into loads on production resources.

2. Automatically generate requirements for personnel.

3. Automatically identify overload conditions at the available resource level.

4. Provide capability to interactively evaluate alternatives (.....) for resolving overload conditions.

(ISMC, III, 11)

A.2.5.2 <u>To Retrieve Automatically the Inventory Requirements</u>

As a minimum Capacity Planning must have a capability which provides the automatic retrieval of inventory requirements and process planning data needed to convert production requirements into loads on production resources.

(ISMC, III, 11)

A.2.6 <u>Production Facilities Loading Needs</u>

A.2.6.1 <u>To Access Automatically the Tool and Material Files</u>

To Include the Following (Minimum) Capabilities:

As a minimum the capability is required to:

1. Automatically verify availability of tools and materials for each production order, and generate shortage list.

2. Automatically call for delivery of required tools and materials when selected load is released to the dispatch function.

3. Automatically generate initial shop floor load based on inventory requirements and tool/material availability.

4. Automatically identify bottleneck conditions of individual resources.

5. Provide for interactive intervention to undo bottlenecks, or to maintain full loading of resources which are of high value.

6. Provide simulation capability for use in interactive intervention.

(ISMC, III, 12)

A.2.7 Dispatch Load Needs

A.2.7.1 To Maintain, On Line, the Following Information

The following information required for this decision must be stored on line with real-time access.

1. Set-up and run time for each machine (.....)

2. Status of each machine, automatically incorporated into history files (loaded with operator ID, out of service, current set-up - with code or other ID, availability, and any other machine data required).

3. Operator(s) available.

(ISMC, III, 12)

A.2.7.2 To Perform the Dispatch Function Manually or Automatically

The dispatch function can be either automatic or manual, a simulation/optimization capability to assist in assigning jobs and operators to machines required for either approach.

(ISMC, III, 12)

A.2.7.3 To Access Automatically Process Performance Data

To Maintain the Following (Minimum) Information:
The following information is required as a minimum

1. Actual set-up and run times, machine used, operator, and disposition of output. (.....)

2. Amount and cause of down time for each machine. (.....)

3. Amount of maintenance performed on each machine.

4. Time spent with no work assigned to each machine.

5. Any "significant" performance among operators.

(ISMC, III, 13)

A.2.7.4 To Agglomerate the Above Information

(The above information) must be amendable to agglomeration in order to arrive at appropriate performance data across the entire center, e.g. total span time for a specific order across the center.

(ISMC, III, 13)

A.3 Test Bed Needs

A.3.1 Integration Needs

A.3.1.1 Integration Demonstration Needs

1. The ISMC is intended to demonstrate the validity of the basic Integrated Application concept. Integration as well as to show specific payoffs resulting from incorporating various tools to affect integration.

   (ISMC, III, 2)

2. Demonstrating various systems engineering tools and techniques in the design and construction of a truly integrated center is equally important.

   (ISMC, III, 2)

3. The key concept being demonstrated is integration.

   (ISMC, II, 3)

A.3.1.2 Integration Definition

Everyone using a piece of data....is exactly using the same data, and further that the data is accurate at the time it is used.

(ISMC, II, 3)

A.3.1.3 Integration Scope

It has proven helpful to consider responsiveness to change and integration from the viewpoint of 4 major groups or threads of functions.

1. Product Technology
2. Control
3. Information Management
4. System Development

A.3.1.4 Excessive Data Redundancy:

Excessive Redundancy of Data Exists

(ISMC, III, 15)

A.3.2 Interfacing Needs

A.3.2.1 Interfacing Stand Alone CAD & CAM Systems

Interfacing stand alone CAD & CAM systems. It is desirable that this CAD/CAM interface be made automatic.
(ISMC, III, 6)

A.3.2.2 Interfacing Standards

Recommendations regarding interface standards and the use and integration of manufacturing applications programs such as process planning and manufacturing control language program generation should be considered.

(ISMC, III, 6)

A.3.2.3 Interface to Process Planning

The ISMC must contain an interface to the Process Planning activity of the technical thread to provide direct access and transfer of the logical data generated in Process Planning.

(ISMC, III, 5)

A.3.2.4 Machine to Machine Interface Problems

Machine to machine interface problems are encountered in some type of data transfers. Programs, user and system messages transactions (Application Processes), data elements and files must be transferrable between similar or unlike processors.

(ISMC, III, 14)

A.3.3 Data Management Needs

A.3.3.1 Data Structuring Needs

1. It is required that the process plan database information be structured and stored to provide access and retrieval of those process plans having specific characteristics.

   (ISMC, III, 8)

2. Process Planning data must be structured and stored to support the operation of numerical control processes by on-line communication of information to numerical controlled processes and control functions.

   (ISMC, III, 9)

3. Data used by the applications must conform to a set of standards and structure to be defined prior to the detailed design of the applications.

   (ISMC, III, 22)

A.3.3.2 Data Portability Needs

Installations are tied to a particular vendor's data support system which dictates and generally restricts the range of options for data structuring and the possible access mates.
(ISMC, III, 15)

A.3.3.3 Data Validation Needs

There are inadequate facilities for enforcement of constraint checks during initial entry of data.

(ISMC, III, 15)

A.3.3.4 Data Integrity Needs

Data is inaccurate and untimely because it is not collected at the source.

(ISMC, III, 15)

A.3.3.5 Data Redundancy Elimination Needs

Elimination of Data Duplication Needs Excessive Redundancy if Data Exists.

(ISMC, III, 15)

A.3.3.6 Time Dependency of Data Needs

Most systems have inadequate provision for retention and control of time dependent versions of data.

(ISMC, III, 15)

A.3.3.7 Data Security Needs

Tight security over permission to access and update data is not generally provided in today's system.

(ISMC, III, 15)

A.3.3.8 Active Data Dictionary Needs

Most data dictionaries are not comprehensive. They do not include all the relevant entity and attribute classes. They tend to be used as static repositories of documentary information rather than as dynamic mechanisms used in the actual accessing and processing of data.

(ISMC, III, 15)

A.3.4 Distributed Database Needs

A.3.4.1 Distributed Databases

Concept explicitly set forth in Figure 2, Figure 3, and Figure 4 appearing in ISMC pages 21, 23, 24.

A.3.4.2 Database CODASYL Compatibility Needs

Database management systems used by the ISMC applications are assumed to be CODASYL compatible.

(ISMC, III, 22)

A.3.4.3 Individual Database Recovery and Backup Needs

Individual database Recovery and Backup Needs providing for concurrent updating, standard data integrity, security, and backup/recovery capabilities.

(ISMC, III, 22)

A.3.5 Heterogeneous Hardware Needs

A.3.5.1 Multiprocessor Heterogeneous Hardware Needs

A heterogeneous processing environment, with some processors being transactions oriented and others supporting primarily batch or mixed batch and on-line interactive applications can be supported.

(ISMC, III, 22)

A.3.5.2 Back-end Database Machine Needs

A back-end database computer or "intelligent" disc drives could also be incorporated at this level (ISMC Mid Level) to improve performance.

(ISMC, III, 24)

A.3.5.3 Local Area Network Needs

The local area network (single continuous cable or interconnected bus or ring architecture) and adoption of a standard protocol facilitate the faster access to all types of data.

(ISMC, III, 22)

"Figure 3 and Figure 4 exhibit local area networks".

A.3.5.4 Local Area Network Standardization & Planning Needs

Lack of overall planning and mangement in the building of communication networks leads to capacity problems, incompatibilities between equipment thus requiring interface "kludges".

(ISMC, III, 14)

A.3.5.5 Local Area Network Expansibility Needs

Inflexibility in types of equipment which can be supported, and inability to upgrade to more powerful units or to make use of new technologies as they become available.

(ISMC, III, 14)

A.3.6 Common Data Control Needs

A.3.6.1 System Usable Directory of Network Characteristics Needs

Most organizations do not maintain an on-line system usable definition and directory of network characteristics. (Capabilities and protocols of each node; primary and alternate access paths).

(ISMC, III, 14)

A.3.6.2 System Usable Directory Needs

Most organizations do not maintain an on-line system usable definition/directory of network characteristics. (Capabilities and protocols of each node; primary and alternate access paths);

(ISMC, III, 14)

A.3.6.3 System Usable Location Directory Needs

(Most organizations do not maintain cf A.3.6.2) security information about legal users; legal transactions (Application Processes) by node; location directory for programs and data, etc.) thus preventing adequate management and control of the system ...

(ISMC, III, 14)

A.3.6.4 Comprehensive Data Dictionary Needs

Most data dictionaries are not comprehensive. They do not include all relevant entity and attribute classes.

(ISMC, III, 15)

A.3.6.5 Active Data Dictionary Needs

(Data dictionaries) tend to be used as static repositories of documentary information rather than as dynamic mechanisms used in the actual accessing and processing of data.

(ISMC, III, 15)

A.3.6.6 ISMC Common Data Needs (Min Configuration)

Additional information defined and brought under centralized control includes: location of data, data relationships, some data constraints, protocols and definition of all "common" data used by ISMC applications.

(ISMC, III, 22)

A.3.6.7 ISMC Common Data Definition Needs

(ISMC) Common data is defined as (1) data used by more than one individual, or (2) data updated by one and used by another or (3) data planned to evolve into a category described in criteria (1) or (2) above or (4) any data which affects information in the above categories.

(ISMC, III, 22)

A.3.6.8 ISMC Common Data Needs (Max Configuration)

Information about the linkages of documentation, procedures, programs, screen formats, reports, user roles and responsibilities, support personnel roles, standards and guidelines are all maintained within the system and can be used as needed to control various functions.

(ISMC, III, 24)

A.3.6.9 ISMC Improved Constraint Checking of Common Data (Max)

ISMC Common Data Growth Version Needs (Max Configuration)

Improved constraint checking will be provided such capabilities as condition checking on existence, non\existence, or specific values of elements related to incoming transaction (Message), in addition to the standard constraint checking of the values of the transaction (Message) parameters.

(ISMC, III, 24)

A.3.7 User Interface Needs

A.3.7.1 User Interface Consistency Needs

Independently developed applications and system utility programs lack consistency in format, command interpretation menu presentation and handling, user prompting, lead through, error diagnostic messages, etc.

(ISMC, III, 13)

A.3.7.2 Users Interface Flexibility Needs

Systems are frequently inflexible and difficult to change. Users cannot get rapid response to changing requirements for information and are dependent upon the data processing staff for all changes.

(ISMC, III, 14)

A.3.7.3 Standard Query/Report Generator Needs (Min)

A standard query/report generator language would be desirable (in the Min ISMC configuration).

(ISMC, III, 24)

A.3.7.4 User Command Language Needs

User Command Language Needs (Mid 82) and the User Command Language(s) must be defined and developed by Mid 82.

(ISMC, III, 24)

A.3.7.5 Standard User Interface Needs (Mid Configuration)

A standard User Interface is provided to insure consistency in user interaction, to provide for standard menu presentation and handling, user help facilities, novice/expert mode of interaction, standard diagnostic messages, access to standard utility packages, etc.

(ISMC, III, 24)

A.3.7.6 User Ad-Hoc Queries Needs (Min Configuration)

Ad-hoc queries and data retrieval across applications can be provided with this configuration. The local area network (....) and adoption of a standard protocol facilitate the faster access to all types of data in the system.

(ISMC, III, 22)

A.3.8 Simulation Needs

A.3.8.1 Manufacturing Planning Simulation Needs

(3) Appropriate Simulations of the proposed center and elements of it which address system configuration, operating strategies, parts spectrum, and performance measures.

(ISMC, III, 7)

A.3.8.2 Master Schedule Planning Simulation Needs

The Master Schedule Planning activity shall have the capability to simulate the effects on production resources, including machine types, personnel, and inventory levels of proposed changes in the production schedule or product mix.

(ISMC, III, 9)

A.3.8.3 Simulation Batch Run Needs

This simulation can be loaded and executed in batch mode, provided that such information as schedules, current resources, ..., span times can be executed.

(ISMC, III, 9)

A.3.8.4 Simulation Implementation: IDSS Needs

"The ISMC is using the simulation capabilities contained in IDSS. This is evidenced on Figures 2, 3, & 4 of the ISMC document".

...... as supported by appropriate tools from IDSS, all running on an information management system similar in capability to the "mid" configuration.....

(Memo, p 1, pp 3)

A.3.9 Processing Needs

A.3.9.1 Database Updating Needs (Min)

At this level, the application programs still control actual updating of the data.

(ISMC, III, 22)

A.3.9.2 Database Updating Needs (Mid)

The IISS will have complete responsibility for the integrity of the data under its control, all updating will be handled by it rather than by application programs.

(ISMC, III, 24)

A.3.9.3 Transaction and Batch Processing Needs (Min)

A heterogeneous processing environment, with some processors being transaction oriented and others supporting primarily batch or mixed batch and on-line interactive applications can be supported.

(ISMC, III, 22)

A.3.9.4 Transaction Prioritization Needs

Lack of dynamic change capability for transaction (Application Process) priorities.

(ISMC, III, 16)

A.3.9.5 Processor Load Leveling Needs (Max Configuration)

An overall automatic network manager is needed to handle load leveling across multiple systems. Typically, load leveling is a manual function, handled by system support personnel; thus it cannot respond to dynamically changing loads.

(ISMC, III, 16)

A network management facility which has the capability to provide some measure of load leveling on processors, provision of alternate routes to data sources and to user device.

(ISMC, III, 24)

A.3.10 Response Time Needs

A.3.10.1 Immediate Retrieval of Engineering Changes Needs

Further, it is essential that the complete status of any engineering change be immediately retrievable, such that affected activities can coordinate their operations with the progress and effects of change.

(ISMC, III, 6)

A.3.10.2 Eight Hour Retrieval of Engineering Change Preparation Data Needs

To support engineering change preparation and other design activities the following data sets must be maintained in a manner such that the most current version of the information can be accessed within a reasonable time, such as eight (8) hours.

(ISMC, III, 6)

A.3.11 Test Bed Resource Usage Statistics Needs

Performance measurement and optimization problems are related to the lack of procedures and mechanisms for the collection and analysis of system usage and performance statistics and the lack of automated techniques for controlling (or assisting the decision-maker responsible for controlling) distribution of processing, distribution of data, communication system load leveling, etc.

(ISMC, III, 17)

A.3.12 Standards and Procedures

A.3.12.1 Test Bed Development Standard Needs

Standards for Integrated Applications (development) (languages, module size limits, structured techniques, system interface rules) must be maintained also.

(ISMC, III, 22)

A.3.12.2 Test Bed Application Standard Needs

Standards which must be utilized in this level of implementation include: DBMS characteristics, data dictionary, communication protocols, the data structure definition (format and content).

(ISMC, III, 22)

A.3.12.3 Test Bed Layering Standard Needs

Standards dealing with architecture layering to provide for the many types of device and data independence refined to in Section A.3.A must also be defined to facilitate upgrading the system and to promote flexibility and responsiveness to change.

(ISMC, III, 24)

A.4 Project Drivers

A.4.1 Responsiveness to Change Needs

A.4.1.1 Responsiveness to Changes in Applications Needs

A mandatory aspect of the ISMC, then, is that all aspects of it be designed for ease of change. The changes can be of several types, for example:

1. Adding shop floor processes, both numbers and types

2. Replacing one manufacturing system with another, which does the same function, for example, replacing a computer aided order release system with one which is fully automated, and

3. Integrating additional manufacturing systems with the ISMC.

(ISMC, II, 2)

It has proven helpful to consider responsiveness to change and integration from the viewpoint of four (4) major groups or thread functions. These are:

1. Product technology

2. Control

3. Information management

4. System development

(ISMC, III, 3)

A.4.1.2 Responsiveness to Change in Data Needs

Systems are frequently inflexible and difficult to change. Users cannot get rapid response to changing requirements for information and are dependent upon the data processing staff for all changes.

(ISMC, III, 14)

A.4.1.3 Responsiveness to Change in Equipment and Technology Needs

Inflexibility in types of equipment which can be supported, and inability to upgrade to more powerful units or to make use of new technologies as they become available.

(ISMC, III, 14)

A.4.1.4 Responsiveness to New Data Processing Technology Needs

The solution must be receptive to new information processing technologies.

(ISMC, III, 17)

A.4.1.5 Layered Architecture Needs

This implies an appropriately layered architecture which will allow any layer to be replaced by an equivalent until interchangeable/replaceable IR parts) with minimal disruption to he adjacent layers.

(ISMC, III, 17)

A.4.2 Portability Needs

A.4.2.1 Decoupling Logical and Physical Data Structure Needs

Most systems tie logical and physical definitions of the data structure together in an inflexible manner. Whenever the physical organization of the data needs to be changed (for performance reasons or to support new types of data), all programs which reference the data must be modified, recompiled and tested.

(ISMC, III, 15)

A.4.2.2 Eliminating Program Peripheral Dependency Needs

Physical control mechanisms for specific peripheral devices are frequently embedded in application program logic, creating inflexible, device dependent systems.

(ISMC, III, 16)

A.4.2.3 Eliminating Dependency on Vendor Data System Needs

Installations are tied to a particular vendor's data support system which dictates and generally restricts the range of options for data structuring and the possible access modes.

(ISMC, III, 15)

A.4.2.4 Virtual Terminal Needs (Mid Configuration)

...A "virtual device terminal" intercepts all messages to and from terminals and converts diverse incoming characteristics into standard formats for presentation to the application software, and also converts standard output from the applications into device specific formats, control characters, etc.

(ISMC, III,24)

A.4.2.5 Migrating the IISS Dictionary to Back End Machine Needs

A back end database computer or "intelligent" disc drives could also be incorporated at this level to improve performance. The IISS dictionary will be more heavily used in active participation in processing all transactions (Messages and Application Processes).

(ISMC, III, 24)

A.4.3 Technology Development Needs

A.4.3.1 Systematic Development of Integration Technology Needs

Systematically develop and demonstrate those combinations of technologies as required to solve the information system integration problem.

(ISMC, III, 17)

A.4.3.2 Evolutionary Technology Development Needs

The solution must be workable in current environments. It must also support an evolutionary implementation of the new IMT technology into the existing hardware, software and particularly the management systems of the environments.

(ISMC, III, 17)

A.5 Project Management Needs

A.5.1 Schedule Needs

The first Test Bed demonstration would occur by 1 Feb 83

(Memo, p 1, paragraph 3)

A.5.2 Test Bed Configuration Needs

It will include the shop floor control system from 6103, integrated with appropriate modules of a commercially available MRP, as supported by appropriate tools from IDSS, all running an information management system similar in capability to the Mid configuration described in our 31 Sept. 81 document.

A.5.3 Test Bed Expected Life Cycle Needs

The Test Bed is designed to readily accept new modules and replacement systems/modules, and the current thinking is for it to remain in operations into 1988 demonstrating new capabilities before they are installed in the ISMC.

(Memo, p. 1, paragraph 3)

A.5.4 Project Traceability Needs

(Lack of) traceability between mission statement (system requirements) and the system specification (exists and must be eliminated).

(ISMC, III, 28)

A.5.5 Test Bed Affordability Needs

The solution strategy must be affordable in a practical sense by the ICAM Program, which implies prioritization with respect to the key technical areas where development is currently lacking in the field.

(ISMC, III, 17)

A.5.6 Technology Transfer to ISMC Needs

The solution strategy must provide proof of tangible benefits by early 1983 and through March 1988 in order to asist in transferring those concepts into the ISMC demonstration.

(ISMC, III, 17)

APPENDIX B

TEST BED USER SCENARIOS (MAXIMUM CONFIGURATION)

*** NOTE ***

The scenarios described in this section attempt to encompass foreseeable use of the Test Bed. They are provided as background information to guide the engineering decisions which will be made during the design phase to ensure maximum flexibility and extensibility of the design. The extent of the implementation of the scenarios described here is dependent upon the level of resources available to the developers of the Test Bed.

B.1 Relationship Between the Test Bed and CBIS

B.1.1 CBIS Definition

CBIS (Computer Based Information System) - A network of information processing resources and associated procedures that enable manufacturing operations and management.

(CBIS 2-1)

B.1.2 Test Bed Definition

A network of information processing resources and associated procedures that enable demonstration of the Information Management Thread, as well as demonstration and validation of Integrated Applications Subsystems.

B.1.3 Discussion

From the above definitions, it follows that the Test Bed is a demonstration CBIS, hence, the Test Bed must satisfy to the functional requirements of CBIS, without satisfying to the operational requirement (number of terminals, availability, response time, etc.) of a CBIS sized to support manufacturing operations and management.

The CBIS class I updates (coordinated updates, performed directly on the databases without involving existing application programs) are excluded from the functionality of the Test Bed. This exclusion reflects the technical complexity and risk implied by the implementation of the class I updates.

B.1.4 Graphic Representation

See Figure B-1.

B.2 Test Bed User Classification

B.2.1 Test Bed User Definition

To be consistent with other technical documents related to the Test Bed, the definition of the Test Bed user includes persons, machines (terminals and real time peripherals), and software modules.

A Test Bed user is

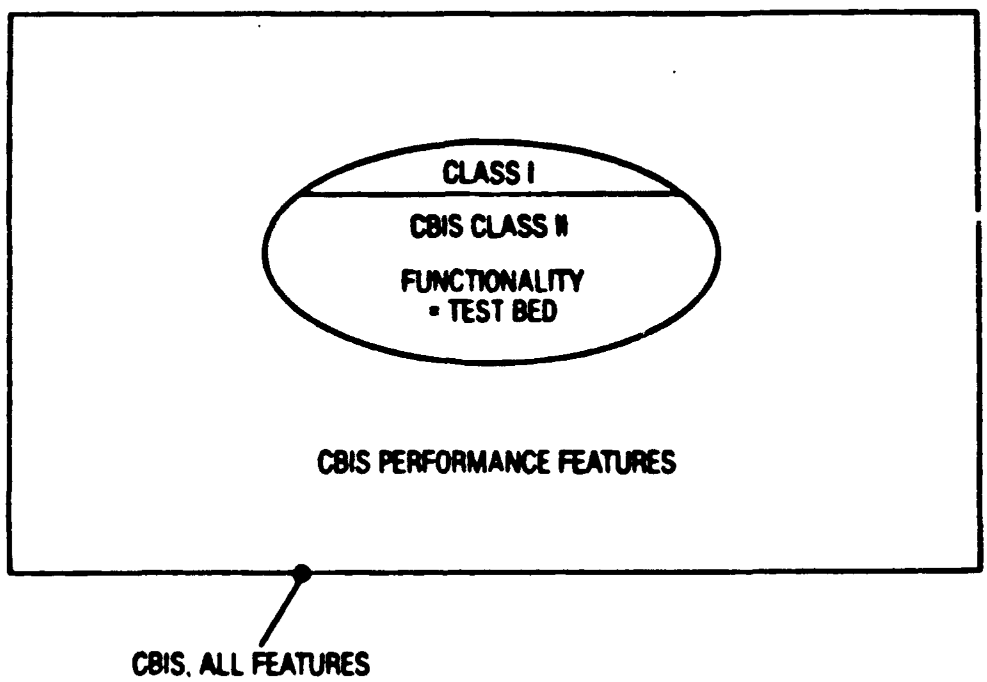

Figure B-1. CBIS/Test Bed Relationship

1. A person, group of persons who makes use of the integrated features provided by the Test Bed.

2. A machine (real time, terminal) which obtains or feeds data to a subsystem integrated by the Test Bed.

3. A software module which communicates with a subsystem integrated by the Test Bed.

A classification of the Test Bed users is shown in Figure B-1. This figure illustrates additional roles which are fulfilled by:

4. Persons who maintain, upgrade, manage or demonstrate the integrated application subsystems or IISS services installed on the Test Bed.

5. Persons who generate and maintain the Test Bed standards.

B.2.2 Test Bed User Role Traceability Matrix

	USER-ROLES	NEEDS OR CBIS REFERENCES					
1.	End User Person		A.3.7	A.3.10			
	Software	A.3.8	A.3.9				
	Machine Hosts	A.3.5.1	A.4.23	CBIS B2			
	Lan	A.3.5.3	A.3.5.4	A.3.5.5			
	NC Machines	A.3.2.4	CBIS A4.7				
	Back-End Data		A.3.5.2	A.4.2.5			
	Machine Terminal	A.4.2.4	A.4.2.2	CBIS A4.6			
	CAD & CAM Systems		A.3.2.1				
2.	CDM-Administrator	A.4.1.2	A.4.1.3	A.3.3	A.3.4	A.3.6	A.3.1.4

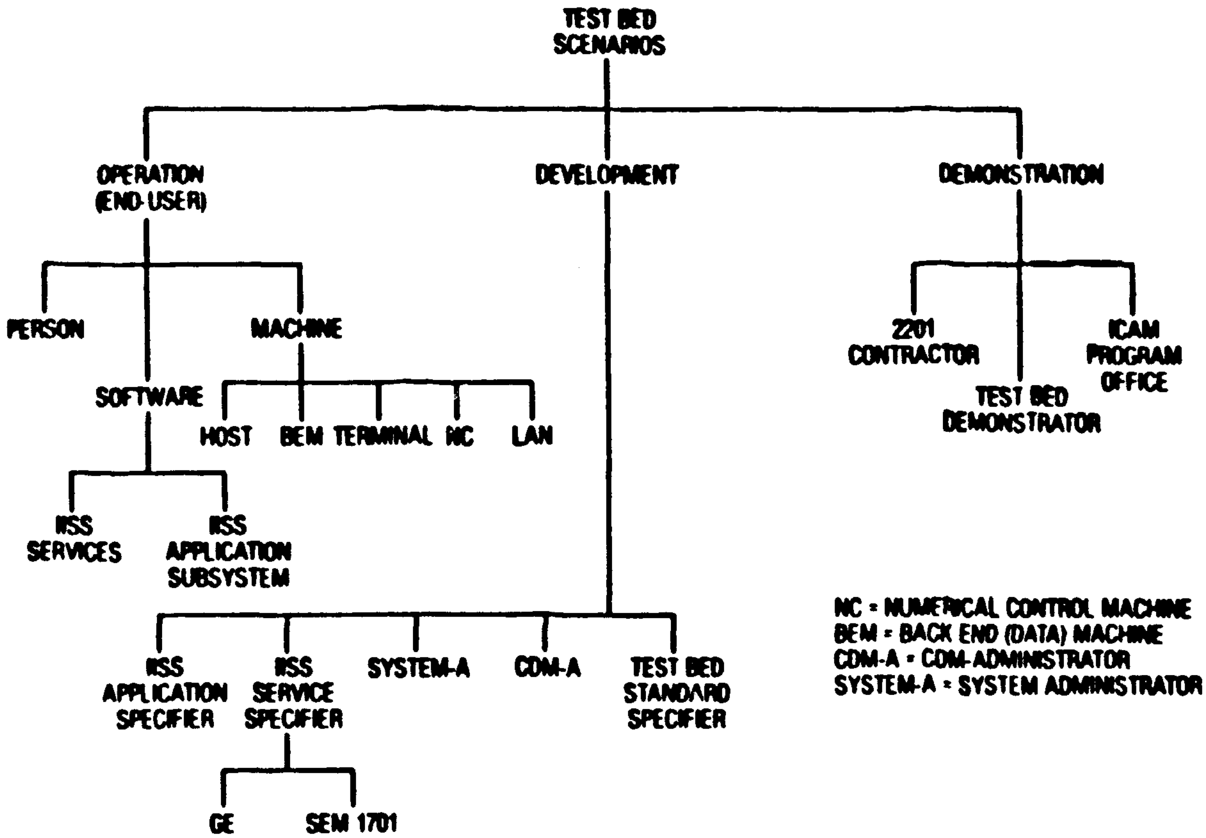

Figure B-2. Test Bed Scenarios Classification

USER-ROLES	NEEDS OR CBIS REFERENCES				
3. System-Administrator	A.3.11	A.4.1.4	A.4.1.5	A.4.1.3	
4. Application Specifier	A.4.1.1	A.4.2.1	A.4.2.2	A.3.2.3	
5. IISS Service Specifier	A.4.1.4	A.4.1.5	A.4.1.3	A.4.2.1	
6. Program Office	A.1.1	A.1.3	A.5	A.1.2.3	
7. 2201 & Other Integrated Applications Contractors	A.1.2	A.3.66	A.3.68	A.3.69	A.2.0
8. Test Developer (not User)	A.4.3	A.3.1	A.5.4		
9. U.S. Air Force	A.1.4				
10. Aerospace Industry	A.1.5				
11. Test Bed Standard & Procedure Specifier	1.3.12	1.3.2.2			
12. Test Bed Demonstrators	A.1.3.2	A.1.3.3	A.1.3.4	A.1.3.7	

B.3 End User Functions

B.3.1 End User Definitions

A person, group or department, a machine tool, etc. which performs a clearly defined manufacturing function. The role performed is defined by the manufacturing organization.

(CBIS 2-1)

B.3.2 End User Role

The End User is restricted in his usage of the Test Bed to invoking any of the following services:

1. Predefined Application Processes
2. Predefined Proceedings
3. Predefined Forms, Help File

4. Available System Services

5. Available Application Process Support Services

6. The IISS Ad Hoc Query Language

7. The IISS User Command Language

The End User plays a role which is fundamentally different from all other roles identified in the Test Bed. He makes use of application processes, procedures and services which already exist.

The application processes, procedures and services are created for him by the Test Bed Application Specifiers. The Application Specifiers respond to the needs of the the End Users by invoking the functional primitives which are provided by the Test Bed.

Although the End User is restricted to a predefined world, a number of services are provided to him. The End User relies on these services to obtain information about the status of the IISS System.

B.3.3 End User Scenarios

Thirty-six (36) End User Scenarios have been identified. These scenarios which are self explanitory are listed in paragraph B.3.4 where they are correlated against the scenarios identified in the CBIS document or in the Needs Analysis (Appendix A).

B.3.4 End User Scenarios Traceability Matrix

END USER SCENARIOS	NEEDS OR CBIS REFERENCES			PRIORITY
1. Sign on/sign off IISS	A.3.3.7	A.3.11		
2. Application Process Directory (request of)	A.3.7.1			
3. Request IISS Status		Verbal request from Program Office		
4. Query status of specific application process	A.3.9.3			
5. Execute predefined message to invoke IISS application	A.3.9.3	A3.1		
6. Execute predefined application process to query IISS data (1 or M databases, files)	A.2.0	A3.1		
7. Execute predefined application process update IISS data (1 or M databases, files)	A.3.9.1	A.3.9.2	A3.1	
8. Execute predefined message to cancel previous application process				
9. Create predefined procedures (made of predefined application processes)				
10. Execute predefined procedures				
11. Delete predefined procedures				
12. Send message to IISS user				
13. Receive message from IISS user				
14. Request up load/down load of binary data	A.3.3.1(6)		CBISA.3.3.7	

B.3.4 End User Scenarios Traceability Matrix (Cont'd)

END USER SCENARIOS	NEEDS OR CBIS REFERENCES	PRIORITY
15. Block predefined application processes in MACRO	A3.2.5	
16. Execute IISS help file		
17. Request IISS file directory		
18. Select output device for application process procedure	A3.2.7	A3.2.6
19. Select input device for application process procedures	A3.2.7	A3.2.6
20. Select output device for system commands		
21. Modify application process priority	A.3.9.4	A.3.2.4
22. Modify own password	A.3.3.7	
23. Run in batch mode	A.3.8.3	A.3.9.3
24. Invoke IISS query language	A.3.7.3	
25. Form ad hoc queries (read-only)	A.3.7.3	A3.2.8
26. Run ad hoc language help file	A.3.7.6	
27. Define data effectivity date, version number	A.3.3.6	A.2.1.4
28. Select novice/expert user dialogs	A.3.7.5	
29. Select application process triggering mode (immediate, deferred, conditional)	CBIS 3-30	
30. Invoke predefined form	A.3.7.3	A.3.6.8
31. Invoke user command language commands	A.3.7.4	
32. Define UCL command files		

END USER SCENARIOS	NEEDS OR CBIS REFERENCES	PRIORITY
33. Execute UCL command files		
34. Delete UCL command files		
35. Invoke report generator	A.3.7.3	
36. Invoke message definition language	A3.2.10	

B.4.0 IISS Services and Application Specifier Functions

B.4.1 IISS Software Classification

Figure B-3 shows a taxonomy of the software run on Test Bed.

Figure B-3 dichotomizes the IISS software into two main categories which are:

1. IISS Applications

2. IISS System Services

This classification leads to recognizing two classes of IISS software specifiers: The system (ISS) specifier and the IISS application specifier.

Table B-1 gives the definition and examples of the six subclasses of software functions shown in Figure B-3.

Table B-2 is the IISS Software Traceability Matrix.

B.4.2 IISS Subsystem Specifier (Application Specifier) Definition

In the above context, the CBIS definition of the application specifier can be retained.

"The mechanism (e.g., person) responsible for translating a manufacturing information requirement into one or more CBIS application processes which satisfy that requirement. The application specifier may also be involved in modifying the Neutral Data Structure, if required.

(CBIS 2-1)

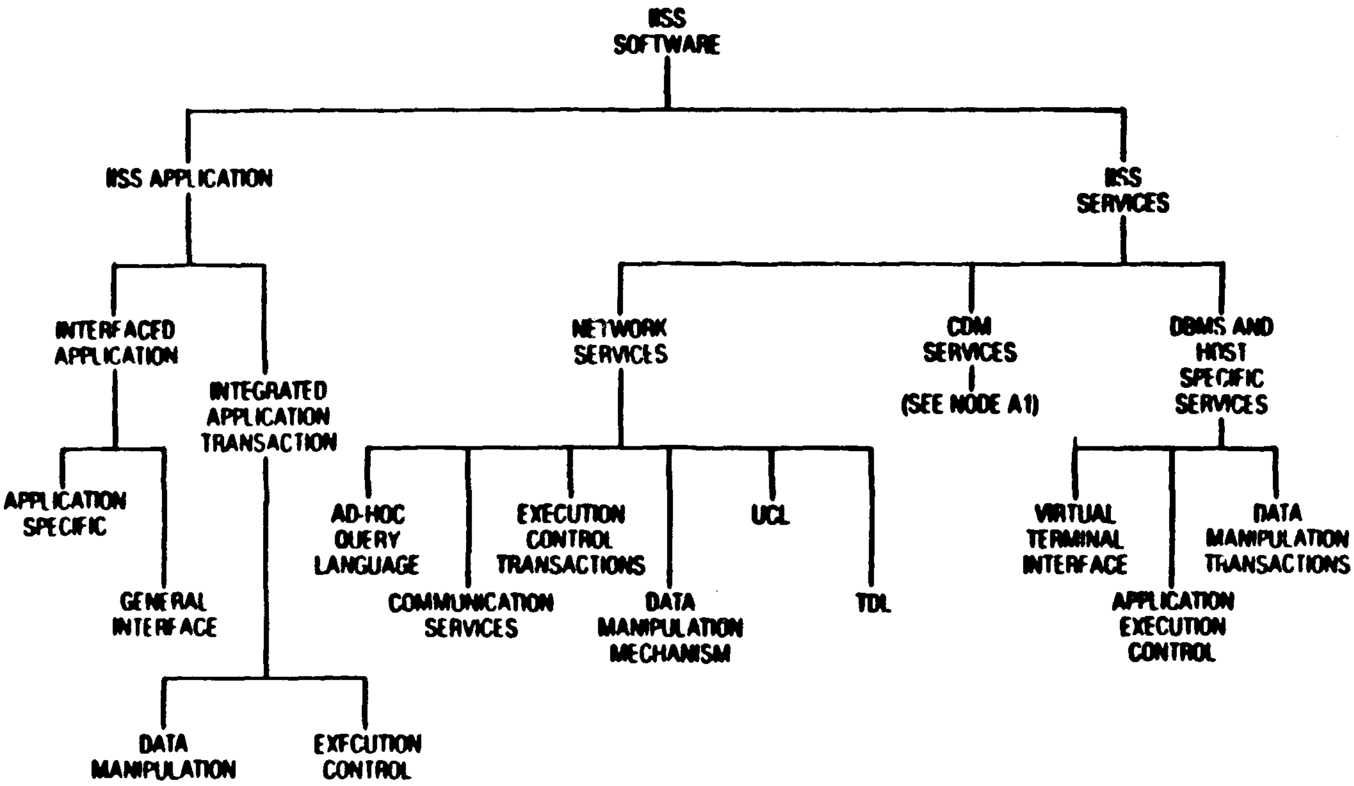

Figure B-3. A0 - IISS Software Classification

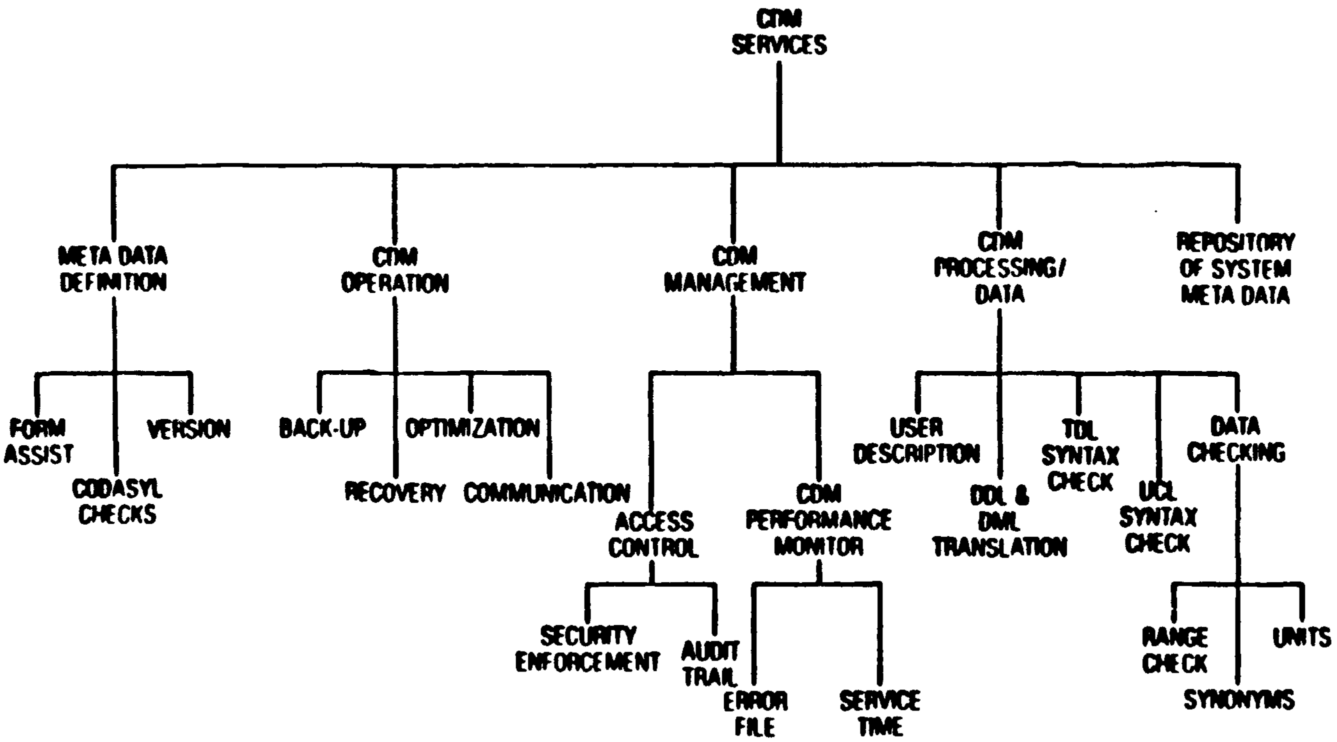

Figure B-4. A1 - CDM Services

TABLE B-1

IISS SOFTWARE APPLICATION DEVELOPMENT DEFINITIONS AND EXAMPLES

1. Stand Alone

   Defined as a non-updating program which makes use of IISS data. Else not within the scope of IISS.
   Example: OR Department wants to make a histogram of flat sheet metal part statistics by surface and by thickness.

2. Interfaced Application

   Defined as a program which is too specialized or too costly to be developed, and which must be used as procured.
   Example: MRP as procured from IBM.

3. Integrated Application

   Defined as a program which obtains some of its input data from the IISS databases and which updates some IISS database(2).
   Example: MC-MM

4. Application Process

   A process which causes specific actions to be taken within IISS. An application process causes either one of the following to happen:

   a. Data Query & Display

   b. Data Update

   An application process is made up from the IISS execution control and data manipulation. Application process primitives supplied by the IISS system developer.

5. Network Services

   IISS services which are made available in a uniform fashion across all nodes.
   Example: Message Guaranteed Delivery

TABLE B-1

IISS SOFTWARE APPLICATION DEVELOPMENT DEFINITIONS AND EXAMPLES (Continued)

6. DBMS and Hardware Specifier IISS Services

   IISS services which interpret and respond to the network IISS services. The DBMS and hardware specific IISS services exist on each mode. These services are OS, hardware, DBMS dependent.

7. CDM Services

   IISS services which provides CDM data to a requestor.
   Example: Message Validation Data

TABLE B-2

TEST BED SOFTWARE CLASSIFICATION - TRACEABILITY MATRIX

	SOFTWARE CLASS	NEEDS ANALYSIS STATEMENTS		
B.1.0	IISS Applications			
B.1.1	Stand Alone Application	Not required. Mentioned for completeness.		
B.1.2	Interfaces Application			
	Application Specifier	A.3.2.3		
	IISS Generalized Interface	A.3.2.2		
B.1.3	Application Process			
	Data Manipulation Procedure	A.3.9.3	CBIS A.4.3.7	
	Execution Control Request	A.3.9.3	A.3.9.4	
B.1.4	Integrated Application	A.3.1.1		
B.2.0	IISS Services			
B.2.1	Network Services			
	Ad Hoc Query Language	A.3.7.6	A.3.7.3	CBIS A3.2.8
	Common Data Model	A.3.6		
	Message Definition Language	CBIS A3.2		
	Local Area Network Protocols			
	Help-Files	A.3.7.5		
	User Command Lanaguage	A.3.7.4		
B.2.2	DBMS and Hardware Specifier Services			
	Virtual Terminal Interface		A.4.2.4	
	Application Execution Control Mechanism			
	Data Manipulation Mechanism			
	Data Query Mechanicm			
	Data Update Mechanism	A.3.9.1	A.3.9.2	

B.4.3 IISS Subsystem Specifier (Application Specifier) Role

B.4.3.1 Application Specifier Role Description

The application specifier is charged with developing, modifying, deleting new or existing IISS applications or IISS application processes.

While developing or modifying an IISS application process, the application specifier may be called to specify or to retrieve the meta-data contained in the CDM.

The application specifier develops new application processes by assembling existing system services to fulfill reliably and rapidly the processing requirements of a manufacturing information application.

B.4.3.2 Application Specifier Non-Role Description

It is not the mission of the application specifier to develop, modify or delete new or existing IISS services. These activities are the responsibilities of the IISS service developer.

The application specifier has the undisputed needs to have access to the CDM and has the knowledge required to specify or update elements of the CDM. However, to maintain centralized control over the CDM it is recommended that the application specifier be only authorized to make temporary changes to the CDM for test purposes. The actual installation in permanent form of these changes should only be done by the CDM Administrator or his deputies.

B.4.4 IISS Subsystem Specifier (Application Specifier) Scenarios

1. Read only access to the CDM to obtain the definition of existing CDM data elements, procedures, etc.

2. Search the CDM for the existence or non-existence of specific data elements, procedures, etc.

3. Create temporary definition of future CDM elements for test purpose.

4. Recommend modifications, update, deletion, insertion of data elements, procedures, files to the CDM Administrator.

5. Recommend modifications, updates, deletion, addition of IISS services to the system administrator.

6. Create, update existing user forms.

7. Recommend new or modified user focus to the CDM Administrator.

8. Design, code, modify new or existing application programs.

9. Design, code, modify new or existing application processes.

10. Compile, link new or existing application program (batch).

11. Compile, link new or existing application processes.

12. Test new or modified application programs and application process on existing database. Inhibit update to the IISS DBMS' during testing.

13. Install tested application processes or procedures in the Test Bed without Test Bed down time.

14. Install tested application program in the Test Bed without extensive Test Bed down time.

15. Install tested forms, help file, new CDM application processes anddata definition without Test Bed down time, defer effectivity until CDM Administrator review).

16. Create and install the documentation of new or modified application processes in the CDM, without Test Bed down time. Defer effectivity until CDM Administrator approval.

17. Upon request of the CDM Administrator remove existing application programs and application processes meta data from the CDM without Test Bed down time.

18. Upon request of the System Administrator, remove existing application programs and application processes from the IISS system, without Test Bed down time.

B.5 IISS System Specifier

B.5.1 IISS (System) Services Specifier Definition

Person responsible for specifying, developing, modifying documenting, and testing new or existing IISS system services.

B.5.2 IISS System Service Specifier Role

B.5.2.1 IISS System Services Specifier Role Descriptions

The IISS system services specifier specifies, develops, modifies, documents and tests the general purpose services made available by the IISS system to the application specifier.

The IISS services can be divided into two broad classes.

1. Network Services

2. DBMS and Hardware Specifier Services

B.5.2.2 IISS System Service Specifier Non-Role Description

It is not the mission of the IISS system service specifier to develop, modify or delete new or existing IISS application programs. These activities are the responsibilities of the IISS application specifier.

Like the application specifier, the IISS system service specifier has the need to access the CDM meta data, and has the knowledge required to specify or update elements of the CDM. However, to maintain the centralized control over the CDM it is recommended that the system service specifier be only authorized to make temporary changes to the CDM for test purposes. The actual installation in permanent form of these changes should only be done by the CDM Administrator or his deputies.

B.5.3 IISS System Service Specifier Scenarios

1. Read only access to the CDM to obtain the definition of existing CDM data elements, message definitions, network resources.

2. Search CDM for the existence/non-existence of specific data element, procedures, network resources.

3. Create temporary definition of future CDM elements for test purposes.

4. Recommend modifications, updates, deletions, insertions of data elements, procedures, files to the CDM Administrator.

5. Recommend modifications, updates, deletions, insertions of network resources or services to the IISS System Administrator.

6. Design, code, modify new or existing network or hardware specific services.

7. Compile, link, new or modified IISS services on specific hardware.

8. Compile, link new or modified IISS network services.

9. Test new or modified hardware specific IISS services on existing database management system, without down time.

10. Test new or modified IISS distributed services without Test Bed down time.

11. Install tested services without Test Bed down time.

12. Create, install documentation for new or modified IISS services on the CDM. Defer effectivity until CDM Administrator review.

13. Measure the performance of new or modified IISS services.

14. Read only access to the IISS error log file.

15. Trace IISS errors to offending IISS system services. Correct errors if necessary.

16. Upon request from the IISS Administrator remove existing IISS services from the IISS system without Test Bed down time.

B.6 <u>CDM Administrator Scenarios</u>

B.6.1 <u>CDM Administrator Definition</u>

A person within the CBIS who analyzes information requirements and develops the neutral data structure.

CBIS 2-1

B.6.1.1 <u>Remark on the Definition</u>

As stated above, the CDM Administrator definition does not emphasize the enforcement role of the CDM Administrator required to implement the centralized control and management of data contained in the CBIS.

Since centralized data management is the theme of CBIS, the following definition of the CDM Administrator is proposed:

"A person within the CBIS who analyzes information requirements, develops the neutral data structure, enforces centralized control of updates to the neutral data structure, controls access to the meta data and who manages the CDM support resources."

B.6.2 <u>CDM Administrator Role</u>

The CDM Administrator main mission is to ensure that the data contained in the various IISS databases is described in the CDM database, that this description is stable and is well controlled.

The CDM Administrator has the additional mission of managing the CDM resources and of ensuring continuous operation of the CDM.

B.6.3 <u>CDM Administrator Scenarios</u>

(see Figure B-5)

B.6.3.1 Develop the Meta Data

(see Figure B-6)

B.6.3.1.1 Definition of the Meta Data

1. Establish Data Requirements for each IISS subsystem.

2. Understand and model the IISS data resource.

3. Build CODASYL data submodels.

4. Integrate data submodels into existing IISS data models.

5. Describe a data record.
   Record name
   Duplicate authorization
   Access authorization
   Attribute description
   Procedure description

6. Describe a set type.
   Owner record
   Member record
   Set membership rules
   Insertion (auto, manual)
   retention (fixed, mandatory, optional)
   Set selection rules
   key, currency, system
   Set ordering rules
   first, last, next, prior, nth, default
   Duplicate sets

7. Describe an IISS application process from data supplied by application or IISS Service Specifiers.

8. Understand impact of an IISS application process on the IISS data resources from the following new points:

   o Data usage

   o Data origination/utilization

   o Data security

   o System throughput

9. Understand IISS user data needs.

10. Establish IISS user security privileges.

11. Describe UCL commands, including syntax, help file, access privileges from data supplied by the IISS Service Specifier.

12. Describe the neutral DDL and DML commands including syntax, help file, access privileges from data supplied by the IISS Service Specifier.

13. Describe the Ad-hoc query language commands including syntax, help file, access privileges from data supplied by the IISS Service Specifier.

14. Describe the IISS application subsystems (address, protocols, etc.)

15. Describe the various IISS database managers (error codes, DDL and DML features, navigation operations, etc.)

16. Describe the various sequential files used in the IISS system.

17. Describe the terminal characteristics used in the IISS system (address, protocol, etc.)

18. Establish the CDM (or CDMs) logical, physical addresses and access rules.

19. Establish the CDM error file(s) logical, physical addresses and access rules.

20. Describe the various IISS users (priority, access privileges, identification).

21. Grant/Deny IISS user data access privileges.

22. Grant/Deny IISS user meta data access privileges.

B.6.3.1.2 Implement the Meta Data

1. Insert new meta data.

2. Update, delete existing meta data.

3. Access existing meta data by record, set type, application process, etc.

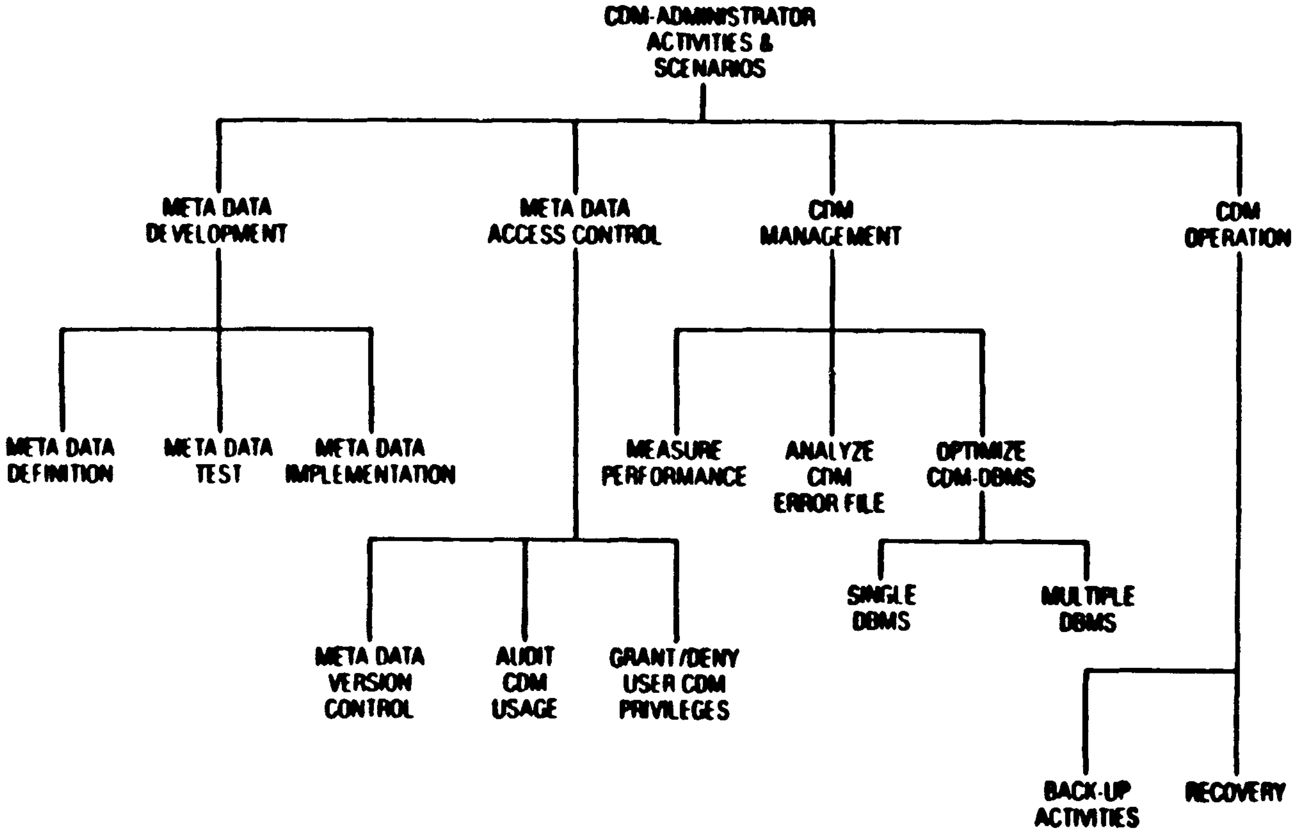

Figure B-5. A0 - CDM Administrator Scenarios

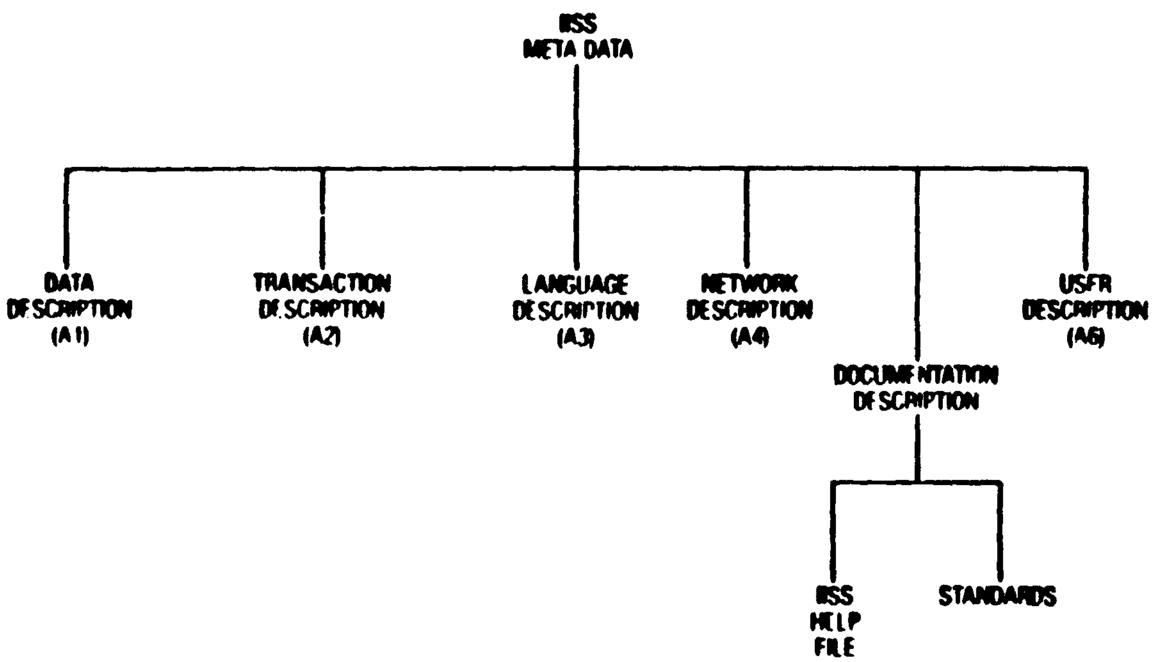

Figure B-6. A0 - CDM Data Description

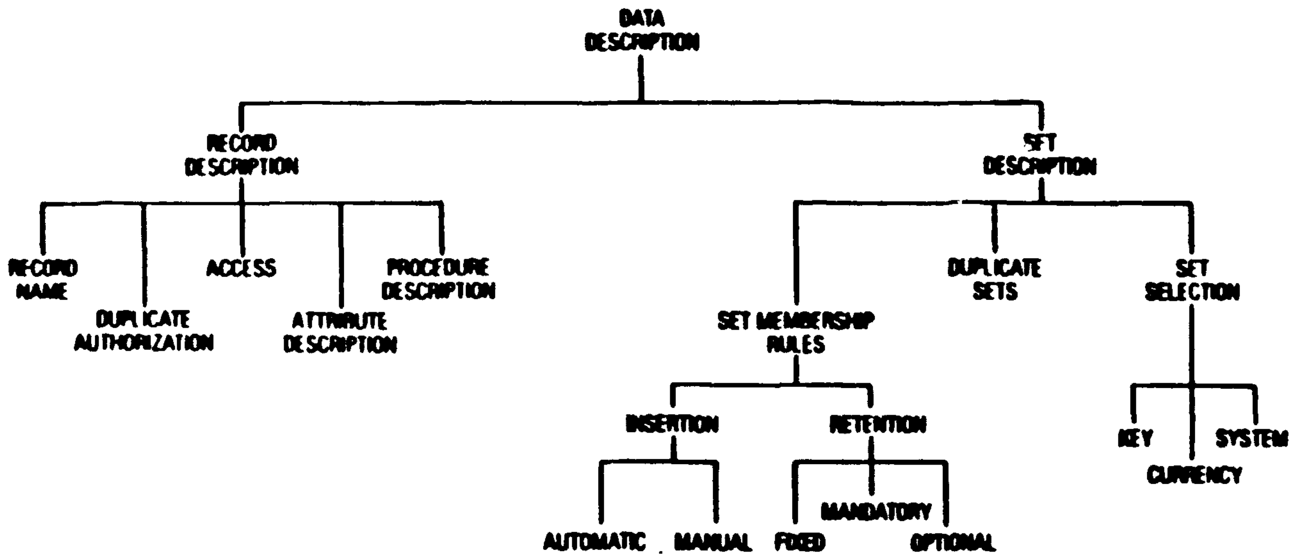

Figure B-7. A1 - CDM Data Description

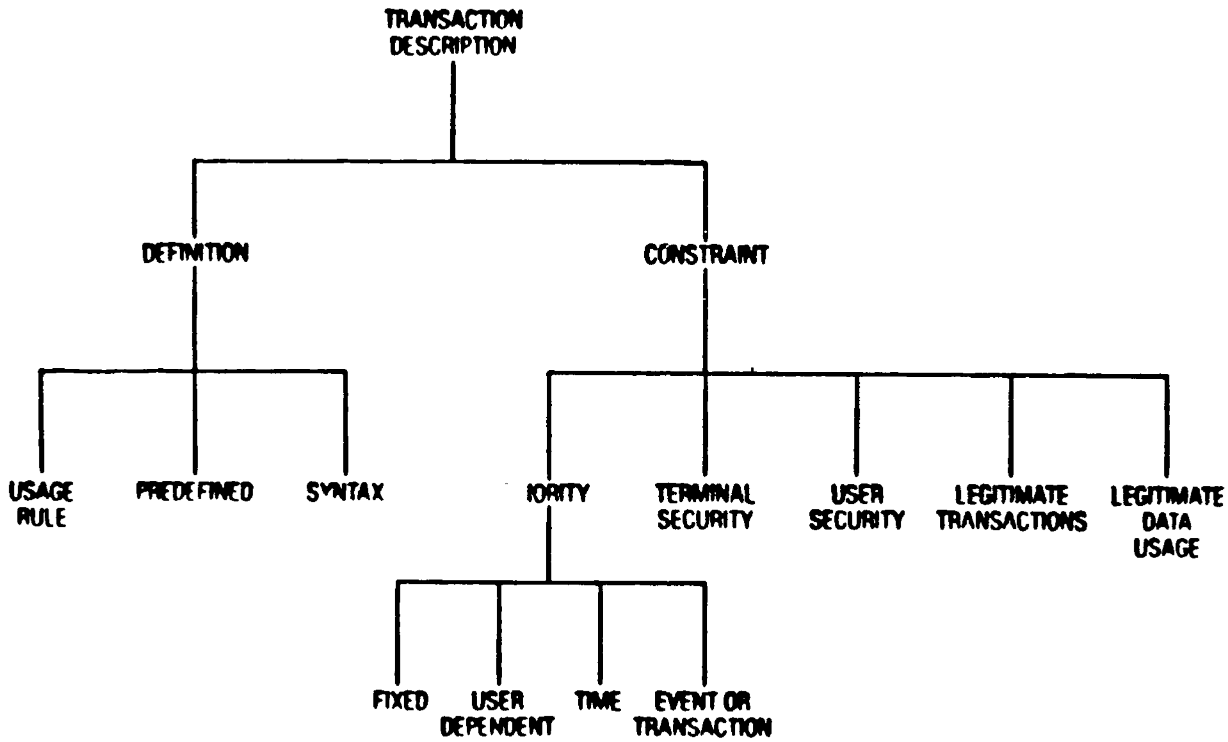

Figure B-8. A2 - CDM Data Description

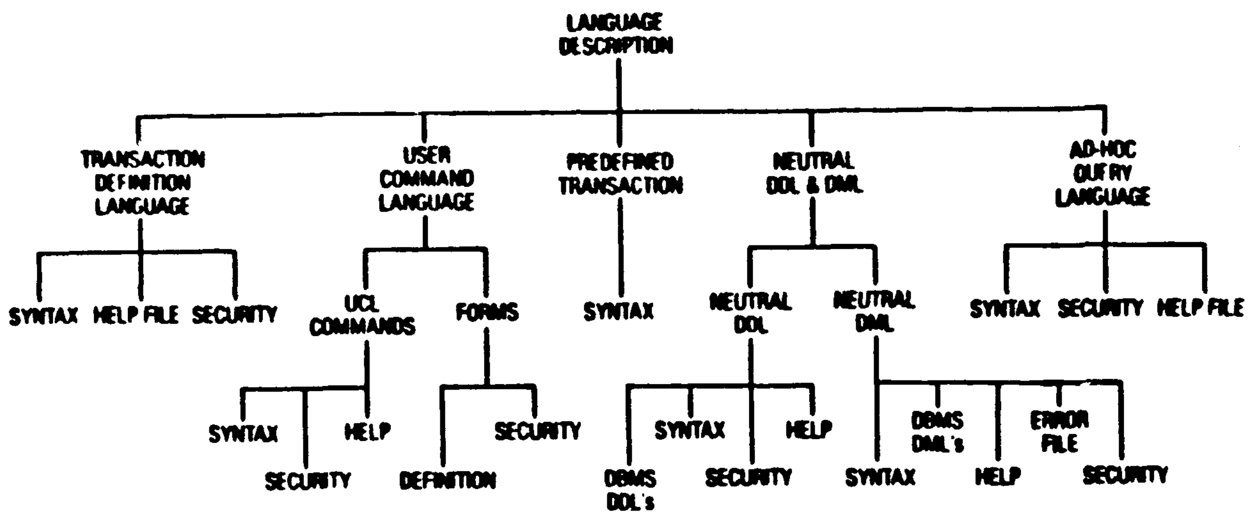

Figure B-9. A3 - CDM Data Description

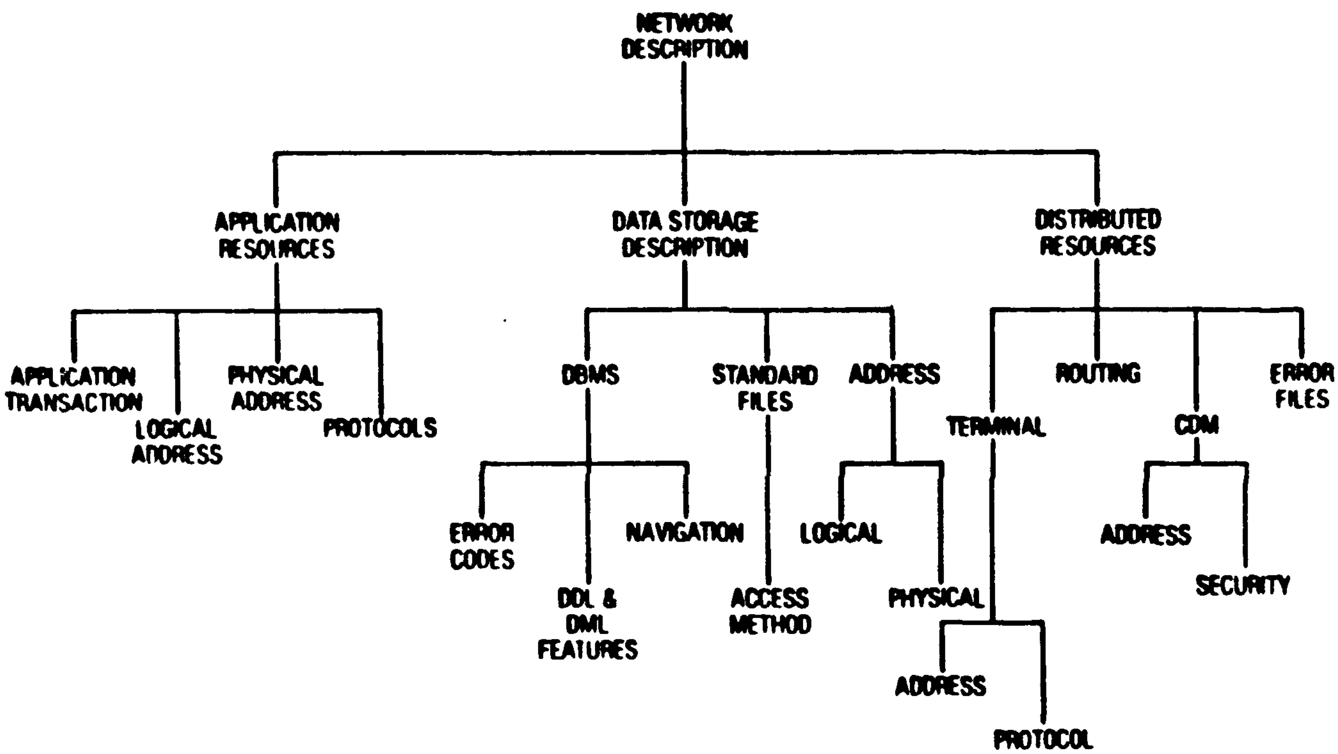

Figure B-10. A4 - CDM Data Description

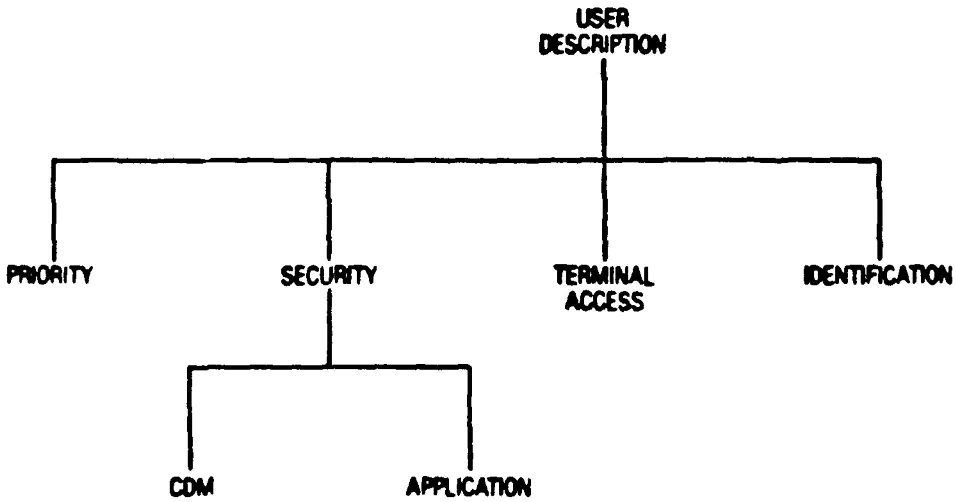

Figure B-11. A6 - CDM Data Description

4. Control meta data revision effectivity.

B.6.3.1.3 Test the Meta Data

1. Create temporary meta data for test purposes.
2. Test for record type name uniqueness.
3. Test for set type name uniqueness.
4. Check syntax of data model against CODASYL syntax.
5. Check ranges and units of data items.

B.6.3.2 Control Access to the Meta Data

B.6.3.2.1 CDM Version Control

1. Establish scope, revision number of CDM data.
2. Create temporary revision to the CDM data, and control effectivity.
3. Automatically update CDM revision number when updates are made to the CDM.

B.6.3.2.2 Grant/Deny CDM Access Privileges

1. Grant/Deny CDM access privileges to specific individuals or groups of individuals.
2. Grant/Deny CDM access privileges to specific application processes or groups of application processes.

B.6.3.2.3 Enforce CDM Security

1. Maintain a CDM access audit trail.
2. Screen audit trail for unauthorized entries.

B.6.3.3 Manage CDM Resources

B.6.3.3.1 Measure CDM Performance

1. Keep running log of CDM accesses by user types.
2. Measure mean CDM service time by user types, application process type.

B.6.3.3.2 Analyze CDM Error File

1. Keep running log of CDM errors.
2. Analyze CDM errors.

3. Resolve CDM errors.

4. Reset CDM error file.

B.6.3.3.3 Optimize CDM Database Management

1. Optimize single CDM database manager

2. Duplicate CDM

B.6.3.4 Operate CDM

1. Back up the meta data.

2. Recover the meta data after a crash.

B.7 IISS System Administrator Role

B.7.1 IISS System Administrator Definition

A person within a CBIS who operate the CBIS Computer System.

(CBIS 2-3)

B.7.1.1 Remark on the Definition

The above definition defines the role of the IISS operator rather than the role of IISS system administrator.

The following definition is proposed: "A person within IISS who administrates and manages the resources allocated to IISS. In particular he controls the level of manpower and hardware resources required by IISS".

B.7.2 IISS System Administrator Role

The main objective of the System Administrator is to provide a system view to the administration of the IISS resources.

The scope of IISS System Administrator includes:

- o Hardware resources
- o Manpower resources (IISS Service Specification & Application Specifier)
- o CDM administration supervision
- o IISS operators

Figure B-12 is an organization diagram for IISS. One individual may fulfill one or several roles.

B.7.3 System Administrator Scenarios

B.7.3.1 Administrator IISS Personnel

1. Direct implementation of IISS services and applications
2. Direct and control IISS service specifier
3. Direct and control IISS application specifier
4. Direct and control the CDM-Administrator
5. Direct and control the IISS operator
6. Enforce implementation of IISS standards
7. Support objectives of the ICAM Program Office
8. Ensure continuous operations
9. Recommend changes to IISS standard to standard specifier

B.7.3.2 Administrate IISS Hardware Resource

1. Audit IISS system usage
2. Grant/Deny IISS user access
3. Audit IISS user response time

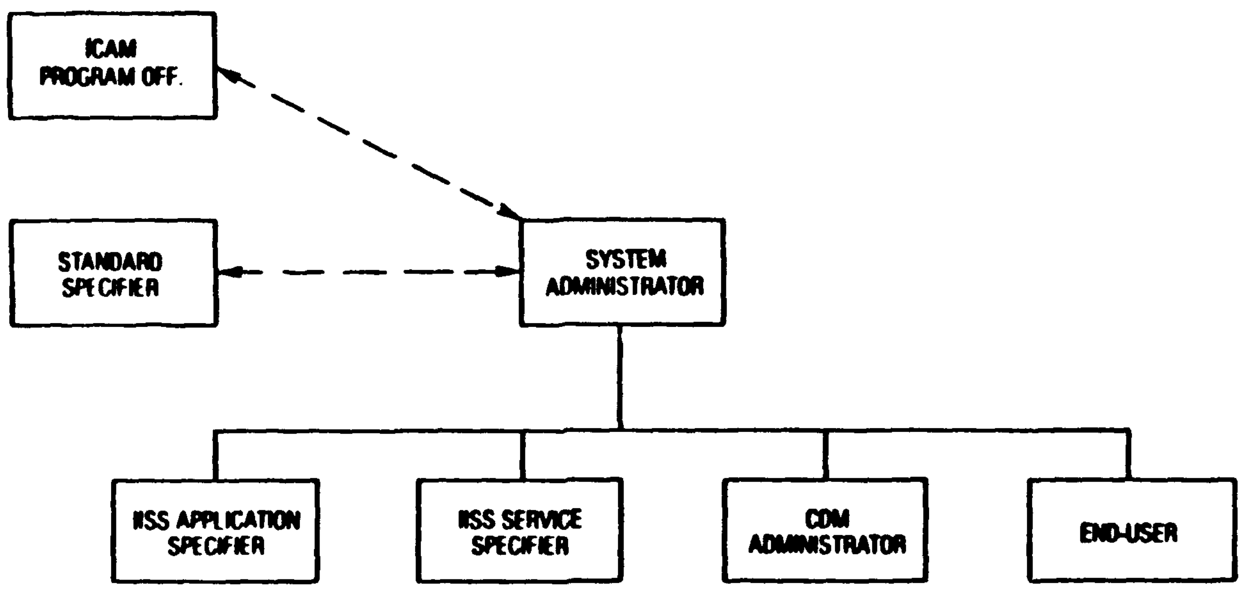

Figure B-12. System Administrator Role

4. Audit IISS hardware performance (LAN, HOSTS, DBMS)

5. Recommend addition/deletion to IISS hardware (LAN, Terminal, host)

6. Recommend duplication of CDM to improve performance

7. Enforce IISS security

B.7.3.3 Perform Extraordinary Recovery Procedures

1. Unlock database accidentally locked by user

2. Boot/restart local area network

B.8 IISS Standard Specifier

B.8.1 IISS Standard Specifier Role Definition

A person who formulates and recommends standards which must be satisfied by IISS service software and by IISS application software.

B.8.2 IISS Standard Specifier Mission Definition

The main objectives of the IISS Standard Specifier mission are to formulate and recommend standards which promote the usefulness of IISS to the users while reducing overall implementation cost.

The IISS Standard Specifier ensure that the IISS standards evolve as the state-of-the-art, state of application in heterogeneous distributed processing evolves.

B.8.3 IISS Standard Specifier Scenarios

1. Keep current with state-of-the-art of IISS support technologies.

2. Keep current with latest Federal, commercial, military, industry standards.

3. Recommend standards to be applied to IISS.

4. Review request for update, modification, deletion of existing standards.

5. Propose update, modification, deletion of existing standards.

6. Assist CDM Administrator in bringing the standards on-line.

7. Assist IISS service specifier and application specifier in implementing standards recommendation.

8. Perform audit of IISS software for standard compliance at the request of the IISS System Administrator.

B.9 DAPro Program Office

B.9.1 DAPro Program Office Role

The DAPro Program Office guides the development of the Test Bed to support the objectives of the DAPro Program and to be of maximum value to the ISMC contractor.

B.9.2 DAPro Program Office Scenarios

B.9.2.1 Definition of Integration (inserted here at the request of the Air Force)

B.9.2.1.1 "Data Integration"

B.9.2.2 Scenarios

1. Systematic development of integration technologies.

2. Provide visibility into cost and problems of integration.

3. Demonstrate flexibility of the integration environment.

4. Demonstrate advanced concepts in information management.

5. Demonstrate that integration is workable in an evolutionary mode.

6. Support all development and integration.

7. Simplify the task of the contractor.

8. Validate subsystems before implementation on ISMC.

9. Identify, quantify initial performance improvements.

10. Provide proof of tangible benefits by early 1983 and enhancements through March 1988.

11. Run Integrated Application system on Test Bed before implementation

12. Provide second verification and validation to integrated software.

13. Educate the Integrated Applications community to the total ICAM picture.

B.10 Integrated Applications Contractor and Integrated Applications Community

B.10.1 The ISMC Contractor

Aerospace company selected by the Program Office to implement the ISMC. The ISMC relies heavily on the Information Management thread implemented by the Test Bed.

B.10.2 ISMC Contractor Role

The ISMC contractor is taking advantage of the integration technologies developed in the Test Bed project to support the implementation of the ISMC.

B.10.3 ISMC Contractor Scenarios

B.11 Test Bed Demonstrators

B.11.1 Test Bed Demonstrator Definition

Persons who will demonstrate the features and capabilities of the Test Bed to the Integrated Applications community, the Program Office and the ISMC (2201) contractor.

B.11.2 Demonstration of Basic Technologies

The demonstration environment includes:

- o Heterogeneous computer hardware
  - -- VAX 11/780, and IBM 30XX
- o Heterogeneous database management system
  - -- IDS-II, IDMS, VAX-11 DBMS, IMS, TOTAL, & ORACLE
- o Local area network

1. Demonstrate simple test programs acquiring data from one or many databases.
2. Demonstrate simple test programs updating data on one or many databases.
3. Demonstrate test program execution control via IISS messages.
4. Demonstrate cancel/abort function with roll back to clean points.
5. Demonstrate generalized interfacing capabilities.
6. Demonstrate user access control via the CDM.
7. Demonstrate data check control via the CDM.

8. Demonstrate data modeling checks via the CDM.

9. Demonstrate LAN data transfer.

10. Demonstrate virtual terminal capabilities.

11. Demonstrate preplanned application processes.

12. Demonstrate user command language commands.

13. Demonstrate features of interest to the Program Office.

14. Demonstrate features of interest to the ISMC contractor.

15. Demonstrate batch processing.

16. Bench Mark System against known stand alone application.

B.11.3 Demonstrate the Integration of MRP, MCMM, IDSS

1. Implement selected manufacturing scenarios

2. Demonstrate preplanned application processes requiring multi-base queries.

3. Demonstrate preplanned application processes requiring multi-base updates.

4. Conduct the demonstration of interest to the Air Force Program Office.

5. Demonstrate Test Bed features of interest to the ISMC contractor.

B.12 Test Bed Software

B.12.1 Test Bed Software Classification

B.12.2 CDM Scenarios

B.12.2.1 Meta Data Input Functions

1. Provide interactive forms to enter/update/delete meta data

2. Provide controlled access to enter/update/delete functions.

3. Provide CODASYL consistency checks to meta data definition.

4. Provide access to meta data by meta data item name.

5. Provide automatic revision number to meta data.

6. Provide temporary update of meta data without effectivity (test mode).

B.12.2.2 CDM Maintenance

1. Provide backup facility.

2. Provide physical CDM data independence.

3. Provide CDM tuning capabilities.

B.12.2.3 CDM Access Control

1. Provide meta data access control to users.

2. Maintain meta data access audit trail.

B.12.2.4 CDM Performance Monitoring

1. Log CDM errors in error file.

2. Provide controlled access to CDM error file.

3. Provide selective reset capabilities of CDM error file.

4. Maintain CDM access statistics (service time by user)

5. Provide controlled access to CDM access statistics file.

6. Provide selective reset capabilities to CDM access stat file.

B.12.2.5 CDM Processing

1. Translate DBMS specific DML call into neutral DML format.

2. Translate DML call in neutral format to target DML format.

3. Translate DBMS specific DDL call into neutral DDL format.

4. Translate neutral DDL call into target DDL format.

5. Supply message verification data upon request.

6. Supply user verification data upon request.

7. Supply data conversion data (units) upon request.

8. Log CDM errors in CDM error file.

9. Down-load selected CDM information on cold start to hosts.
10. Control access to the meta data.
11. Provide data item records, set types description upon request.
12. Provide CODASYL consistency checks of the meta data.
13. Provide synonyms conversion.
14. Supply data items range check data upon request.
15. Provide ad hoc query language syntax checking.
16. Provide error code conversion between IISS DBMs

B.12.3 IISS Distributed Services

B.12.3.1 Transaction Processing

1. Provide processing of predefined system services.
2. Support execution of predefined procedures.
3. Support cancel/abort application processes.
4. Support subsystem execution control message.
5. Support batch mode subsystem execution control.
6. Support prioritized application processes.
7. Log user application process by user, application, time stamp.
8. Support application process directory by user.
9. Support selection of output device for application process.
10. Create/delete/update application process command file.
11. Execute an application process command file.
12. Test for application process completion status.
13. Support begin application process function.
14. Support end application process function.
15. Support message definition language.
16. Invoke other application processes.

17. Transaction Definition Lanuguage (TDL) to be set oriented.

18. TDL to be recursive.

19. TDL to contain text editor.

20. TDL to be non-procedural.

21. TDL to be as rich as PL/1.

22. Execute an application process in test mode.

B.12.3.2 Ad hoc Query Language

1. Provide ad hoc query language to query the various IISS databases.

2. Provide ad hoc query language to query sequential IISS data files.

3. Provide ad hoc query language command files.

4. Provide ad hoc query language help file.

5. Provide access control to the ad hoc query language.

6. Provide ad hoc query syntax checking.

7. Provide ad hoc query language functional expension.

8. Invoke ad hoc query processing.

9. Exit ad hoc query processing

B.12.3.3 Communication Services

1. Support information transfer between two subsystems.

2. Support information transfer between subsystem and the CDM.

3. Support transparent information transfer between two users.

4. Provide guaranteed deli ry of data update requests.

5. Provide logging of data update requests for each relevant process.

6. Provide logging of communication errors.

7. Support predefined terminals and application subsystem protocols.

8. Accept and provide data to and from real time users.

9. Control user access to communication services.
10. Synchronize all IISS clocks.
11. Obtain terminal and application subsystem protocols from the CDM.
12. Transmit/receive mail between two IISS users.
13. Download/upload binary data and programs.
14. Recover a down node.

B.12.3.4 IISS System Services

1. IISS log in function with user access control enforcement.
2. IISS log out function.
3. IISS bye function.
4. IISS application accounting by user.
5. IISS help function.
6. IISS file directory.
7. IISS application process directory.
8. IISS cancel/abort function.
9. IISS system and application status.

B.12.4 IISS Hardware and DBMS Specific Functions

B.12.4.1 Virtual Terminal

1. Support predefined terminal protocols.
2. Support predefined application subsystem protocols.
3. Support binary download/upload.

B.12.4.2 Application Subsystem Execution

1. Support application subsystem execution via local message processor.
2. Support application subsystem cancel/abort via local message processor.
3. Support application subsystem status query via message processor.

4. Send message to another IISS application subsystem (on the same host).

5. Send message to another IISS application subsystem (on another host).

B.12.4.3 Data Update Services (Interfaced or Integrated)

1. Update data contained in a local DBMS.

2. Update data contained in a local sequential file.

3. Update data contained in one or many remote DBMS.

4. Update data contained in one or many remote ISAM files.

5. Perform range checks on data to be updated.

6. Obtain range check data from the CDM.

7. Return range check errors to application program requesting update. Do not update database.

B.12.4.4 Data Query Services (Interfaced or Integrated)

1. Query data contained in a local DBMS.

2. Query data contained in a local sequential file via an application process.

3. Query data contained in one or many remote DBMS via an application process.

4. Query data contained in one or many sequential files via an application process.

5. Check status of a local update application process (DBMS) via a message.

6. Check status of a remote update application process (DBMS) via a message.

7. Check status of a local update application process (Sequential) via a message.

8. Check status of a remote update application process (Sequential) via a message.

9. Perform range checks on data which is retreived.

10. Obtain range check data from the CDM.

B.12.5 Application Subsystem

B.12.5.1 Integrated Application

1. Execute an Integrated Application which queries data contained in either one of:
   - Local DBMS
   - Local sequential file
   - One or many DBMS, local or remote
   - One or many sequential files, local or remote
2. Execute an Integrated Application which updates data contained in either one of:
   - Local DBMS
   - Local sequential file
   - One or many DBMS, local or remote
   - One or many sequential files, local or remote
3. Execute an Integrated Application in batch mode.
4. Cancel a running application subsystem, going back to a clean point.
5. Execute an Integrated Application which contains the following COBOL DML calls or equivalent:

   a. Commit

   b. Connect

   c. Disconnect

   d. Erase

   e. Fetch

   f. Find

   g. Free

   h. Get

   i. Keep

   j. Modify

   k. Order

    l. Ready

    m. Reconnect

    n. Rollback

    o. Store

6. Obtain meta data from CDM.

7. Perform user access control.

8. Obtain status of previously requested query or update action.

9. Obtain IISS system status.

10. Execute an application process command file with framed commands.

11. Perform checks on data obtained from the IISS database.

12. Obtain check data from CDM.

13. Execute an application subsystem in test mode (w/o update on DBMS).

B.12.5.2 <u>Interfaced Application</u>

1. Obtain data via interface mechanism from either one of:

    a. Human user

    b. Machine user

    c. Other software user

2. Obtain interfacing meta data from the CDM.

3. Supply data via interface mechanism to either one of:

    a. Human user

    b. Machine user

    c. Other software user

4. Report interfacing errors to the initiating program.

5. Perform checks on data acquired through interface.

6. Execute interfaced application in test mode (w/o\update to DBMS).

APPENDIX C

TEST BED MIGRATION PATH

C.1 Acknowledgment

This document has been prepared by the General Electric Company and updated by the Control Data Corporation for the DAPro 6203 Project in the scope of activities conducted for the ICAM Program Office under Project Priority 6201M and the DAPro Program Office under Project Priority 6203.

The help and participation received by General Electric from its 6201M Subcontractors and Control Data from its 6203 Sunbcontractors is acknowledged.

C.2 Foreword

This document maps out a migration path for the Test Bed into the foreseeable future. The time horizon considerd in this study has been extended to 1990. It is felt that this end point is reasonable when considering the accelerating rate of change of the Computer Industry. Seven years [document written in 1983 and updated in March 1988] will undoubtedly bring tremendous changes in the economics of hardware and software, which would make longer term predictions unreliable.

This document thus sets out to investigate the likely role, configuration and features of the Test Bed circa 1990. To that effect, the following questions are considered:

1. What will be the role of the Test Bed in the U.S. Aerospace environment?

2. What will be the implementation of the Test Bed in such an environment?

3. How will applications be implemented on the Test Bed in the same environment?

Taking a system view of the Test Bed requires considering the architecture of the computer system (hardware and software) implementing the Test Bed, the methodology and tools required to implement additional application on the Test Bed, and lastly, the data environment of the Test Bed. These three facets of the integration of manufacturing data on the Test Bed are portrayed in Figure C-1.

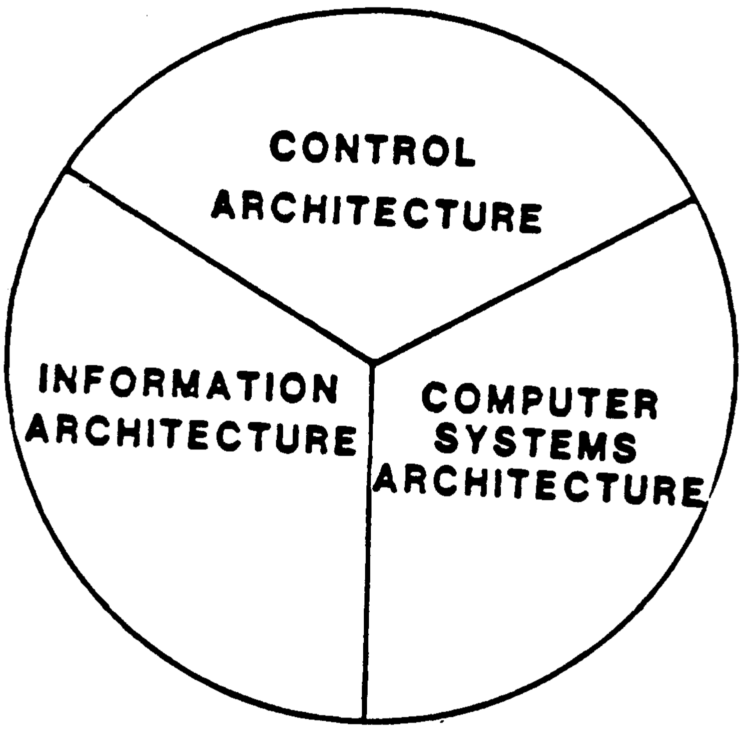

Figure C-1. Environment of the Test Bed

The terms used in Figure C-1: Control Architecture, Computer System Architecture and Information Architecture are defined in the glossary.

Figure C-1 serves as a configuration diagram for this report. The information architecture which is the environment of the Test Bed, is described in sufficient detail to understand the requirements placed on the 1990 Test Bed. This description is contained in the section entitled, "Test Bed Manufacturing Environment". It must be observed, however, that the manufacturing environment of the Test Bed is outside the scope of the Test Bed. It is the forcing function of the Test Bed, rather than a force controlled by the Test Bed itself. The Control Architecture and the Computer Architecture are on the other hand directly affected by the design strategy and by the degree of implementation of the Test Bed. Consequently, this report focuses on these two fundamental aspects of the Test Bed.

For the sake of brevity, the computer system described here, which supports distributed processing, distributed intelligence and integrates new or existing manufacturing databases is referred to as the Test Bed. The rules and procedures used to implement additional applications and their databases on the Test Bed are referred to as the Integration Methodology.

C.3 Glossary

CBIS - Computer Based Information System - A CBIS is a "technology environment" defined as a set of three architectures, which are intended for the management of information within a factory environment. The three architectures include an information architecture, a computer system architecture, and a control architecture. The CBIS interfaces with the "real-time" environment, but it does not contain real-time processors, such as robots and numerically controlled machine tools.

Computer System Architecture - the aspect of a FOF that includes all of the computer hardware, firmware, operating systems, communication facilities, data management facilities, programming languages, and system software necessary to support the physical CBIS.

Control Architecture - the aspect of a CBIS that includes the plans, organizational structure, standards, software engineering methods, procedures, logical data models, cost-tracking mechanisms, etc., necessary for integration of the information architecture and the computer system architecture and for managing their balanced evolution.

Information Architecture - the aspect of the CBIS that includes the logical views of the information, reflecting specific user information needs at a point in time. The information architecture is highly dynamic, since it reflects users' information requirements concerning operational, management and strategic information needs. (See Figure C-5.) It integrates the technical and control threads.

Distributed Intelligence System - Computer System in which small processors at terminal locations carry out functions such as editing, validity checking, User Interface support and communications. (DP MARTIN)

Distributed Processing System - A Computer System in which application programs and/or data reside in separate interlinked processing nodes and are designed in an integrated and tightly controlled fashion. (IBM, 1978)

Back End Processor - Processor dedicated to database management which in general performs this function for one or more computers in a network. (WEITZ)

C.4 Executive Summary

This document describes the migration path most likely to be followed by the Test Bed as it moves from the experimental stage to full production status. The migration path has been established over the 1990 time frame.

This document presents the hardware, software and economic drivers which will shape up the use, structure and development of the Test Bed. Specific milestones are presented in the hardware, software and control dimensions of the Test Bed. A prioritized list of items to be developed in the near term (2 years) is also given, and is shown in the overall context.

C.5 Test Bed Manufacturing Environment

In The Third Wave, Alvin Toffler devotes a full chapter to what he sees will be the manufacturing environment of the years to come. Toffler describes the future manufacturing environment under the caption "Beyond Mass Production" and forecasts an end to the long production run characteristics of todays (perhaps more accurately of the mid 1960's) mass manufacturing. Through well researched statistics, he extends this prediction to the military environment as well, and forecasts shorter and shorter runs for complex, standard weapons. The production runs of the B17 (5,000 +), of the B-52 (600 +), of the future B-1 (200) certainly illustrates this point of view, more surprisingly perhaps, and to quote Toffler, "An eye opening analysis of Pentagon spending by the number of end products purchased came up with the finding that out of $9.1 billion spend on goods for which the number of end items was identifiable, fully 78 percent ($7.1 billion) went for goods produced in lots of under 100 units".

In the civilian sector, as in the military sector of U.S. manufacturing, the need arises for manufacturing complex, high quality products at competitive prices with short production start up time. In the military environment, short production start up time is of particular importance to the fast changing threats posed by the ever increasing sophistication of the adversaries of the United States.

Thus, the move is on, away from the mass production of standardized products to the small lot production of custom products.

The main obstacle to this transformation is the information and decision making explosion implied in the effective manufacturing of custom products. In this environment, product and process information needs to accompany the work in process on a lot by lot basis. Furthermore, manufacturing decision making must be made on a lot by lot basis as well, since the manufacturing process needs to be customized according to product requirements. In short, such a production environment has moved away from a process environment to that of a job shop. In the past, job shops have not been able to enjoy the "economies of scale" associated with mass production. This is the challenge of the Test Bed which integrates product information, process information and decision making/support capabilities. If successful, the Test Bed integrated product and process data resources coupled with advanced decision making and support capabilities will bring full customization on a continuous flow basis. In such an environment, machines can be automatically reprogrammed so that the units of output, each one different from the next, stream continuously from the machines in an unbroken flow.

The above illustrates the key contribution of the Test Bed to the U.S. Aerospace Industry and by consequence to the U.S. Air Force. Stating clearly the objective is a necessary but not sufficient condition for success. A computer system can only be successful if the hardware, software architectures and implementation policies are commensurate to the task. The following sections of this report set the stage for the likely changes in the hardware and software cost drivers over the forecast period (1990) and propose a migration path for the hardware, software and for the Test Bed Control Architecture.

C.6 Test Bed Computer System Architecture

C.6.1 Test Bed Hardware Environment

The hardware environment of the Test Bed of the 1990 reflects the following considerations:

1. Increasingly large number of end-users

   One major study indicates that after 1985, 70% of the U.S. working force would work with computers for at least some portion of their work activity.

2. Microprocessor Economics

The availability of small personal computers is evident in all walks of life. The microprocessors have proliferated because of their low cost, and the resulting low cost per machine instruction they afford make them ideally suited for application programs which require only 8 or 16 bit arithmetic.

3. Information Storage Economics

The rapid advances of VLSI technology will change the storage economics as megabit storage chips become available. Four megabit bubble chips have already been developed by Intel (1982). Moore's Law, which links the number of storage cells per circuit to time, has consistently predicted a doubling of storage capacity every year since 1964. In spite of these rapid advances, Moore's Law predicts some future for the rotating mass memories.

4. Software Path Length

The operating systems of large general purpose computers are increasingly complex and add a very significant overhead to any application program.

The number of operating system instructions executed in support of an application program is referred to as the software path length. Studies have shown that on large main frames, the path length often exceeds 100,000 instructions, whereas on smaller machines (mini's) the software path length is often less than 2,000 instructions.

5. Advances in Teleprocessing

Local Area Networks have brought about a quantum jump in simplicity, capacity and reduced cost with which computers can be interconnected. The current intense standardization activities in this area will further ease the interconnection of heterogeneous computers via off the shelf DMA interfaces to the LAN. The involvement of companies such as IBM, Texas Instruments, DEC, Intel and Xerox guarantees that widely accepted LAN standards will coexist and that gateways between standards will be available. Wide Area Teleprocessing with increased capacity and reduced cost is also a certainty.

C.6.2 Test Bed Hardware Architecture

The hardware architecture shown on Figure C-2 represents the architecture of a production Test Bed suitable for processing in the Aerospace Manufacturing Environment. This architecture offers the following key features:

1. Local Area Network

A high performance, commercially available, Local Area Network is used to interconnect the various computer systems shown on Figure C-2. The Local Area Network is characterized by:

- o High data rates (in excess of 10 megabits)
- o Network configuration capabilities
- o Network resource management capabilities (errors, data rates)
- o Commercial availability of gateways to other LAN

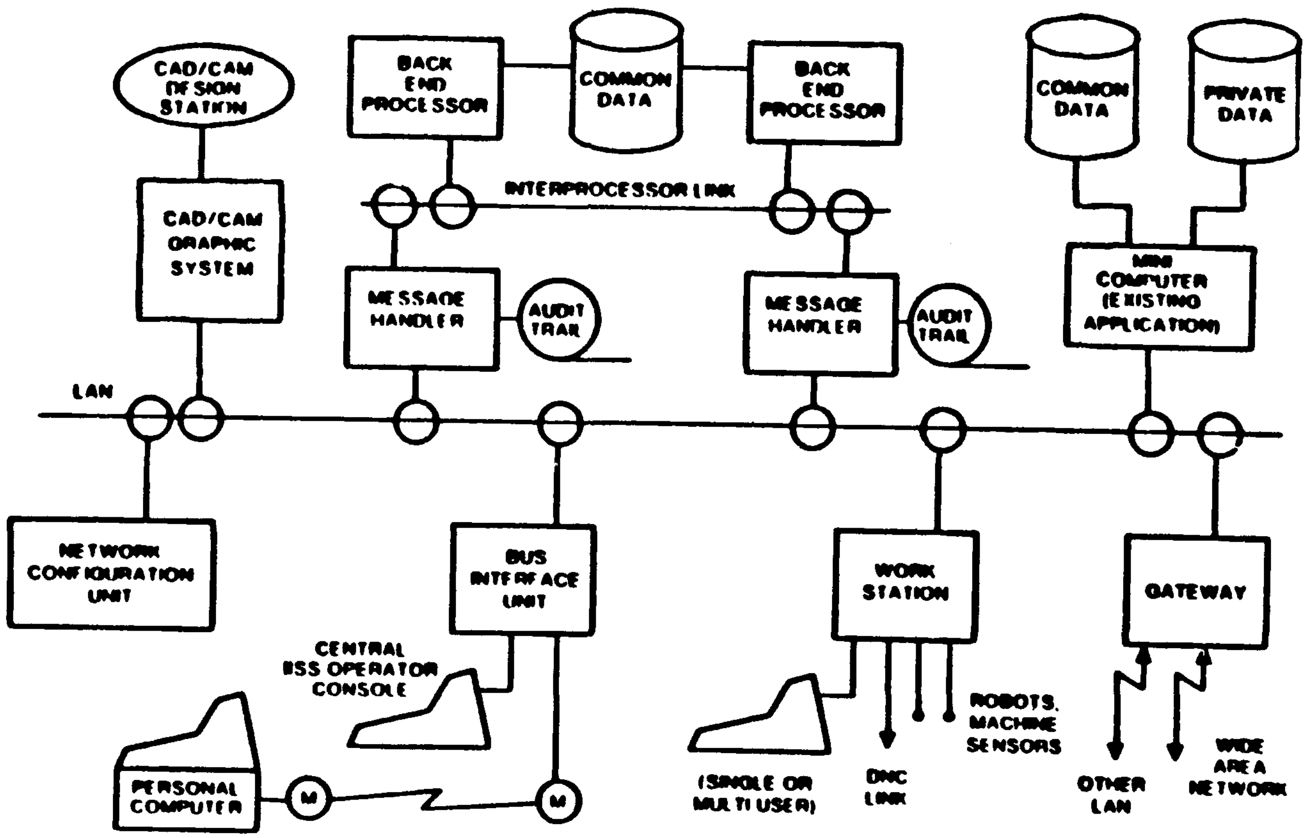

Figure C-2. Test Bed Hardware Architecture and WAN (Wide Area Networks)

2. Local Area Network Interfaces

Two types of interfaces to the Local Area Network are shown on Figure C-2. The first type, already available in 1982, is the Bus Interface Unit. These units allow the interconnection of RS 232 devices to the LAN. Typically, these units are microprocessor driven and support multiple (dozen or so) RS 232 devices. The second type, now emerging from the product literature, is a direct interface between the LAN and the various computer systems shown on Figure C-2. Such interface is characterized by high data transfer rates and by direct interfacing with the computer bus itself rather than through a serial I/O port. Direct interfacing with the LAN will become increasingly more economical as VLSI LAN interfaces become available. Such products have already been announced, [D. ELLIOTT]. The same VLSI interfaces will also allow the direct connection of workstations to the LAN.

Rapid advances in fiberoptics interconnection and interface technology coupled to the favorable price trend in the fiberoptics itself allow the forecast of a shift to fiberoptics, away from coaxial cables. Fiberoptics have significant advantages over coaxial cables: isolation, high throughput, excellent electromagnetic immunity, and are inherently difficult to tap (thereby increasing the security of the network). The key to capitalizing on the VLSI LAN interfaces is the selection of a widely accepted LAN standard and the selection of "popular" computers and work stations.

3. Mini-Computer

The hardware architecture of Figure C-2 can interconnect a large number of mini-computers and other processors. The mini-computers need not be entirely dedicated to the Test Bed. They are used to support the manufacturing application programs now in existence and their databases. These mini-computers will also support the Test Bed services (NTM, COMM) required for the integration of these databases. In the long run, however, new application programs are most likely to reside on the work stations. Such an architecture would maximize parallel processing and lessen the load of the minis, thereby speeding up accesses to the databases stored on the mini-computers.

4. Back End Processors

The back end processors shown on Figure C-2 perform the accesses to common data required by the integrated and distributed processing of the Test Bed. The back end processors will be used to support the databases containing common (shared) data and the CDM. The use of the back end processor for these tasks is cost effective and provides added security to the data.

Back end processors enjoy the following advantages over general purpose computers: 1) simpler and more performing software (shorter software path length), 2) favorable cost ratio (1 to 2) in 1982, and 3) greater security: The absence of general purpose programs on the back end processor makes it easier to audit the data usage made in the system. There are no application programs executing on the dedicated back end processor, thereby eliminating the threat of malevolent programs monitoring database activity (WEITZ). As of this writing, the speed advantage claimed by back end database manufacturers has failed to materialize. This speed advantage is unlikely to materialize as long as the back end processor uses the same mass storage technology as its general purpose cousin.

5. Message Handler Processors

The Test Bed relies heavily on the processing of messages for its coordination and control. Dedicated message handler processors are shown on Figure C-2 to carry out the time consuming message handling operations required for the reliable processing of Test Bed messages. This approach further off loads the back end processors (and could also be applied to the mini-computers of the Test Bed) and speeds up processing. This approach has already been used successfully by the Bank of America in its on-line banking system in California (WEITZ). The message handling processors could implement part of the NTM functionality.

6. Redundant Hardware

As the Test Bed moves into production, the demand for one hundred percent availability will arise. This is because the Test Bed will become effectively the eyes, ears and brains of the manufacturing environment. Redundant hardware provides a means for reaching (with a high probability of success) the 100% availability goal since single failures do not bring the system to a halt. Redundant hardware also facilitates the scheduling of the maintenance operations. Figure C-2 shows how redundant hardware can be connected in a cross over arrangement. An interprocessor link - high speed, vendor supported - is used to switch over the back end processors and message handlers. Such equipment is commercially available. Hardware redundancy should be used on the shared portion of the system critical to Test Bed operations.

7. CAD/CAM Graphic System

As product information becomes increasingly more intertwined with process information required to run the factory, graphic systems will be brought into the Test Bed. Such addition will greatly facilitate process planning and the programming of robots. Direct interfacing of the turn key, commercial graphic systems will be possible. Strong market forces and the emergence of standard high performance VLSI LAN interfaces make this development unavoidable by the CAD/CAM vendors.

8. Workstation

Single or multiple user work stations provide computer power to the user in a cost effective fashion. Work stations are characterized by short software paths, sufficient computing power (16 bit floating point) and memory to support the User Interface functions (screens, application processes, etc.) associated with the Test Bed. The trends in RAM storage technologies allow to forecast the availability of 1 to 4 megabits of solid state RAM storage for the work stations. Because of their computational capabilities, work stations distribute the intelligence of the Test Bed and support near real time machine sensors.

These sensors allow the monitoring of robots, NC machines and other process parameters. Programs run on the workstations are down loaded from the rotating mass memories via the LAN. The distributed intelligence of the workstations will increase significantly the throughput of the Test Bed by relieving to a maximum extent the processors performing the data access operations.

9. Dial Up Capabilities

The Bus Interface Units, are part of the hardware architecture of the 1990 Test Bed. These units allow for the time shared and dedicated interfacing of terminals and personal computers. Personal computers can thus be temporarily connected to the Test Bed via dial up modem. In this mode, a personal computer becomes a non-dedicated work station of the Test Bed.

C.6.3 <u>Test Bed Software Environment</u>

The Software Environment of the Test Bed of the 1990 reflects the following considerations:

1. People and Computer Economics

   The year 1979 marked the crossover point of the computer and people cost trend lines, with people cost now exceeding computer cost in most DP installations.

2. Increasing Share of Purchased Software

   The application backlog now existing in most dp installations and the increasing cost of a debugged line of software make purchased software that much more desirable. It has been estimated that by 1983, 20% of application software will be purchased, up from 9% in 1980 [ADMARTIN].

3. Stern Resistance to Changing Existing Applications

   The high cost of a debugged line of code ($10 per line of COBOL on the average, 1980) and the existing development backlog deter DP installations from making changes to running application programs.

4. Push for Data Processing Productivity

   The data processing departments are under increasing pressures to improve the productivity of their programming staffs. In general, this push results in a strong preference for program acquisition in lieu of program development. Program acquisition has several aspects: it can be accomplished via non-procedural languages, very high level languages, shared code (system services) and database management systems.

5. Fewer Professional Programmers

   End users are becoming more and more knowledgeable in computer science and are increasingly desirous and capable of developing application programs quickly without the tedious and laborious interfacing with the professional programming staffs, well versed in computer science but ignorant of applications.

6. Database Management

   Great productivity improvements are available when application programs are developed for use with a database management system. Furthermore, major application programs are, for this economic efficiency reason, based on commercially available database management systems.

7. Distributed Data Processing

   The above considerations must be viewed in the context of Distributed Processing. Distributed Data Processing problem is a current state of research topic, with no full fledged commercial solution to the Distributed database Management problem.

C.6.4 Test Bed Software Architecture

The Software Architecture of the Test Bed of the 1990 reflects the concernsfor people productivity and for the efficient use of computer resources. In particular, the software architecture allows the distribution of intelligenceamong the various work stations. This minimizes the load placed on the processors having the burden of processing the various data access operations.

1. Predefined and Ad-Hoc Capabilities

   Predefined and ad-hoc distributed query and update capabilities coexist on the Test Bed for efficiency reasons.

2. Layered Software Structure

   Figure C-3 shows the layered structure of the Test Bed Software. Figure C-3 illustrates a Test Bed integrating three different databases supported by three different database managers. Figure C-3 depicts the predefined and ad-hoc capabilities.

3. Distributed Software Architecture

   The distribution of the Test Bed Software across the mini-computers, the workstations and the back end data machines allows significant improvements in processing performance. The functional distribution of the software may take on several forms, depending on the availability of the hardware, and on the location of existing applications in the network.

   a. Initial Distribution of the Test Bed Software

      Figure C-4 shows the initial breakdown of the Test Bed Software into independent functional modules. For simplicity, the Test Bed Services are not shown on Figure C-4. By reference to this figure, it can be seen that some level of parallel processing can be achieved. However, in this arrangement the User Interface functions and the application program are supported by the processors performing the data access operations.

   b. Future Distribution of the Test Bed Software

      Future distribution of the Test Bed Software, Figure C-5, shows the future distribution of the Test Bed Software into independent software modules. This figure assumes that a workstation of sufficient computing power is available. In the predefined query/update environment, a work station of minimum sophistication is sufficient. In the ad-hoc environment, more computing power and memory are required to support the parsing and decomposition of the user requests into single node transactions. Figure C-5 maximizes parallel

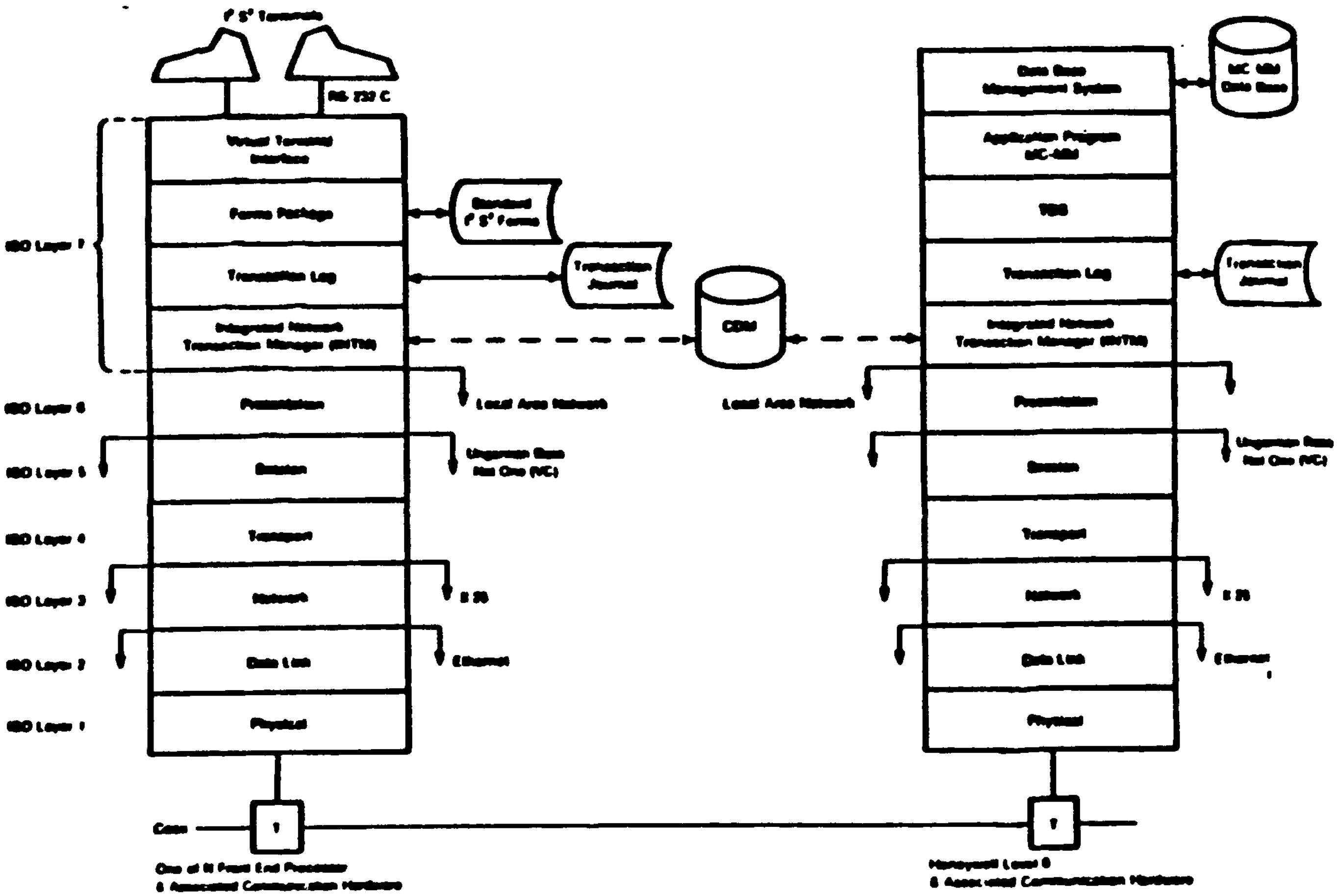

Figure C-3. Test Bed Software Architecture

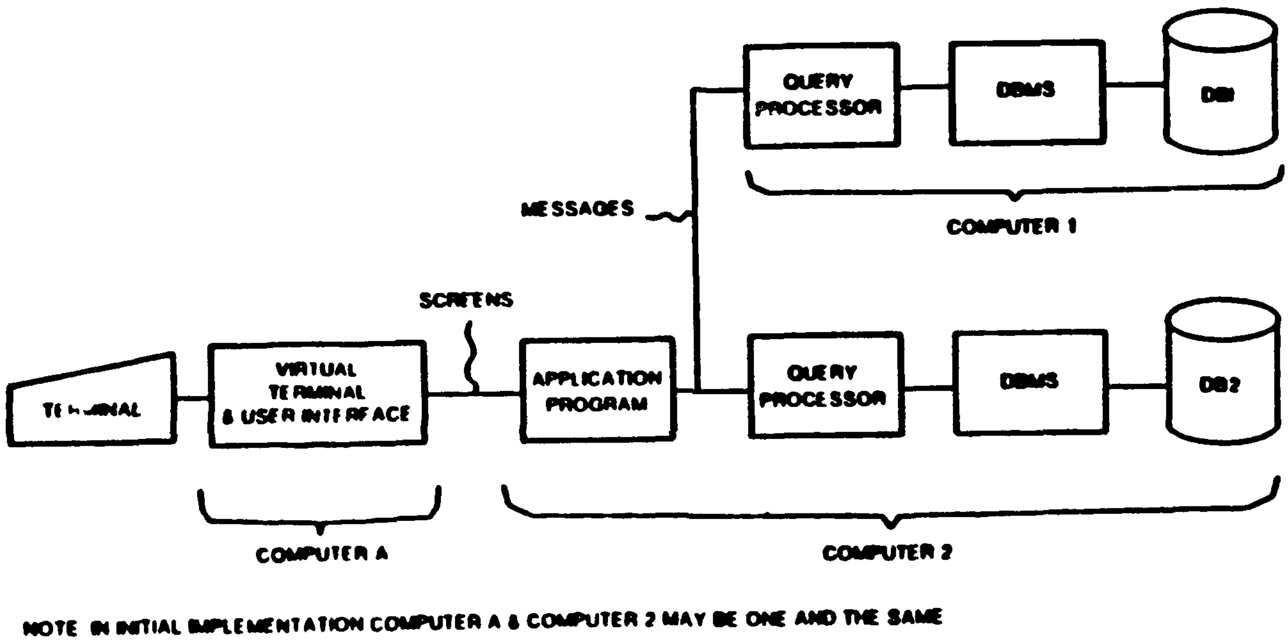

Figure C-4. Initial Test Bed Software Distribution

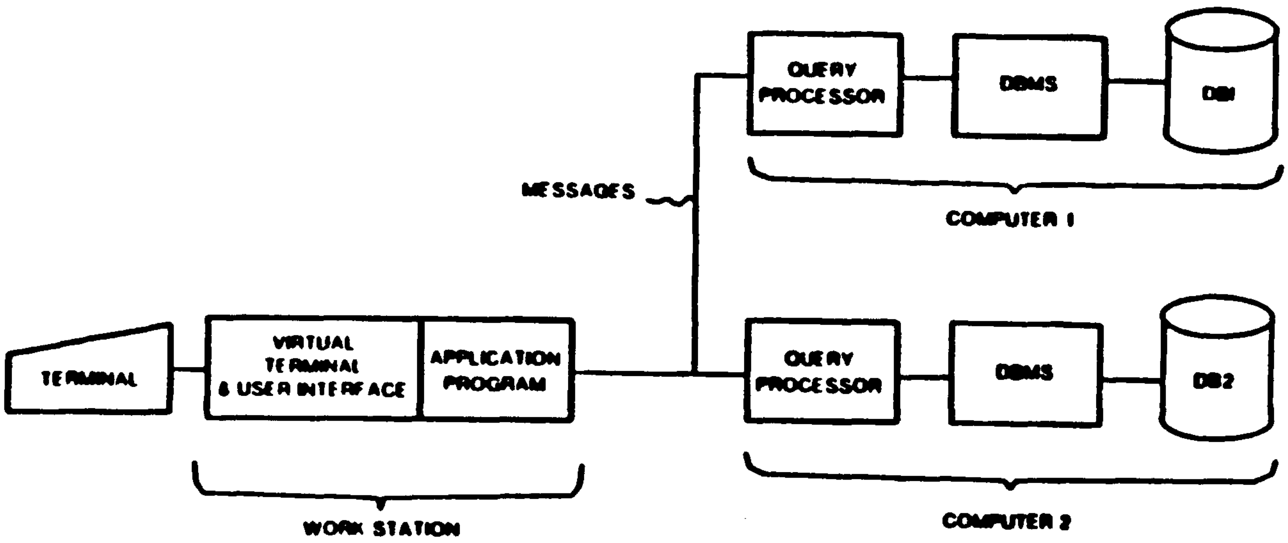

Figure C-5. Future Test Bed Software Distribution

processing and frees to a maximum extent possible the processors performing the data access operations.

This arrangement, it must be noted, accommodates a large number of users in the ad-hoc environment, since the overhead related to the ad-hoc processing is maximally distributed.

4. Shared Code and Services

To maximize programmer productivity the Test Bed allows the sharing of existing code. This applies to system services such as user screens and data integrity checking software. Transaction processors already written and tested are shared among users.

5. Report Generator

A report generator facilitates the accessibility of the data by the user.

6. Higher Level Language

A Higher Level Language allows the non-procedural definition of user specified processing. This language offers control capabilities and defines macro operations of special interest to the manufacturing end user. These macro operations represent frequent and well defined operations in the manufacturing environment. Moving tools from the tool crib to the manufacturing cell or station represent an example of a typical macro operator routinely performed by the Test Bed users.

7. Integration of Non-Codasyl Databases

The Test Bed allows the integration of non-Codasyl databases. Of particular interest to the U.S. Aerospace Environment are the hierarchical and indexed sequential database structures. Specific instances of these database structures will progressively be integrated on the Test Bed by extending to these DBMS the Neutral Data Manipulation Language of the Test Bed. IMS, and TOTAL are examples of the DBMS of interest.

8. Intelligent Retrieval from Databases

Artificial Intelligence Techniques are becoming available to carry out intelligent data retrievals from databases. Of particular importance to the manufacturing environment is the fulfillment of a query which requires knowledge not explicitly stored in the database but which is common knowledge among the users of the system. The representation and usage of the required common knowledge requires Artificial Intelligence Methodology. [NILS].

9. Natural Language Processing

Although relational algebra permits the exact definition of ad-hoc query, in a concise, non-procedural format, it does suffer from a lack of user-friendliness. Natural Language processors provide a bridge between the user and the query processors by performing the semantic analysis of the user natural query and its transposition into its relational algebra. ON-LINE-ENGLISH is such a natural language processor which is commercially available. The use of a natural language processor in the manufacturing environment is very desirable for obvious reasons [CDATE].

10. Performance Optimization

The initial Test Bed implementation provides a first level of performance optimization. Given a conceptual data retrieval path, the first implementation of the Test Bed minimizes transmission cost based on the actual volume and pathing is performed without optimization. Further performance gains can be realized by optimizing the conceptual and internal paths followed to retrieve the data. Such optimization requires the use of heuristics graph search procedures described in the Artificial Intelligence literature [NILS]. The performance optimization can be used for both predefined queries and for ad-hoc queries.

C.6.5 Test Bed Data Environment

The data environment of the Test Bed of the 1990 reflects the following considerations:

1. Existing Databases

The databases which exist in a given manufacturing environment must be preserved. These databases contain information of vital importance to the entreprise and are tied to a gigantic software investment which cannot be economically altered. These databases are, in general, heterogeneous.

2. Data Integrity

The integration of databases places a premium on data integrity since, in the distributed environment, errors affect unsuspecting users and since errors may propagate in a fashion unknown to the human user.

3. Data Independence

Although reasonably stable, the data resource must be flexible to allow addition and deletion reflecting new operating conditions or level of integration. Changes to the data resources must not be allowed to impact the investment made in the software processing the data.

4. Data Redundancy

Some level of data redundancy must be allowed to improve system performance and availability. Data redundancy is, however, a thorny issue since it impacts adversely on data integrity and complicates significantly the update logic. In the Test Bed, some level of redundancy is inherited from existing databases and must be dealt with.

5. Data Relatability

Addition to the data resource must be related to the data already existing in the data resource. The new addition blends in and can be manipulated by the same tools and to the same extent than the original data.

6. Data Accessibility

The data contained in the databases integrated by the Test Bed must be easily accessible by the authorized users. Data accessed frequently is accessed via predefined and precompiled transactions. Data accessed less frequently is accessed via ad-hoc queries. Precompiled transactions and precompiled queries must coexist.

7. Data Shareability

The data contained in the databases integrated by the Test Bed must be shared between several users. The Test Bed must provide the environment where simultaneous data can be retrieved and updated without interference between the various users.

8. Data Security

Adequate level of data security must be provided to the Test Bed data. Data security must be commensurate with the threats of loss of confidentiality since the more secure data security precautions are invariably the more costly of system resources.

9. Data Effectivity

Management of the data life cycle is an important aspect consideration for manufacturing data. Manufacturing data evolves accordingly to a well defined life cycle: preliminary, design, fabrication, maintenance, archive. The ability to treat each datum accordingly to its position in the data life cycle is required to support Aerospace Manufacturing.

10. Data Typing

Well defined data types are provided. The definition of the data types includes the definition of the domains and of the legitimate operations that can be applied to those domains. The enforcement of the constraints on the defined data types are provided by shared system services.

C.7 Test Bed Control Architecture

The control architecture of the Test Bed provides the methodology and the tools required to implement new applications and databases on the Test Bed. The methodology and tools provided allow for the addition and modification to the data structure which supports the Test Bed. In short, the control architecture deals with the creation and maintenance of the Test Bed Common Data Model.

For the Test Bed to be implementable in the production environment, the integration methodology used to integrate databases must be clearly established and further it must be automated.

1. Application Development

   Guidelines (in the form of programming guides) explain how to develop an application process which utilizes the services of the Test Bed (screens, integrity checks, etc.) and which performs distributed query and updates on the databases integrated by the Test Bed.

2. Common Data Model Version Control

   Version control of the Common Data Model is provided. Changes to the Common Data Model are automatically logged. Affected application processes and system services are flagged for recompilation. Application processes and system services obsoleted by changes to the Common Data Model are prevented from being executed by Test Bed services.

3. Computer Assisted IDEF1

   Computer assisted IDEF1 modelling tool allows the generation of normalized IDEF1 models in support of the Test Bed integration methodology and activities. This semi-automated tool allows the interactive checking and building of IDEF1 models. The tool is used to ensure:

   - Uniqueness of key classes as identifiers of entity classes
   - Migration of inherited key classes
   - Compliance with the no null no repeat rule for attribute classes
   - Detailing of relation classes (cardinality, time dependence, boolean, etc. constraints)
   - Detailing of path assertions
   - Detailing attribute class domain constraints

The tool is used in the following scenarios:

- o Construction of an isolated IDEF1 model
- o Construction of an IDEF1 model of an existing database defined by its external schema (which will become an internal schema for the Test Bed)
- o Augmentation of the IDEF1 model to the conceptual schema of the Test Bed

4. Computer Assisted Mapping Definition

A computer assisted tool allows the definition of the External to Conceptual and Conceptual to Internal mappings required to integrate databases and application processes. This tool accepts as input the External and Conceptual schemas already defined and prompts the CDM Administrator to fully define the corresponding mappings. Likewise, the tool accepts the Conceptual and Internal schemas as input and prompts the CDM Administrator to define the Conceptual to Internal mappings.

5. On Line Data Dictionary

An on line data dictionary is automatically maintained from the External, Conceptual and Internal schema definitions. The dictionary is used by the CDM Administrator to keep track of the entity classes, attribute classes, and domain constraints already defined to the Test Bed. Sufficient information is kept about entity classes and attribute classes to unequivocally identify the semantics of each class.

6. Application Process Interference

The data dictionary records the possible interference between the Application Processes. Interference possibility is deducted from the external schemas associated with the application processes. This approach allows concurrent processing of non-interfering application processes and greatly enhance system throughput (SDD-1).

7. Test Bed Resource Management

Resource management capabilities are provided to support the assessment of Test Bed usage patterns. This information is required for the tuning of the Test Bed, and for budgetary control.

8. Data Distribution Strategy

Guidelines assisting in distributing the data optimally in the Test Bed are provided. These guidelines provide the CDCDM Administrator with sufficient information to decide upon the degree of redundancy desirable for optimum system performance as well as the distribution of the data in the various databases integrated by the Test Bed.

9. Augmentation to the Conceptual Schema

A methodology supporting the development and augmentation of the conceptual schema is provided. The methodology provides an IDEFO of the procedures to be followed and an IDEF1 model of the information to be gathered.

10. External Schema Definition

A computer assisted tool is provided to assist the CDM Administrator in building the external schemas used by the various application programs. This tool provides a systematic procedure to the definition of the external schemas built by the user. The items contained in a typical external schema are described on Figure C-6.

11. Internal Schema Definition

A computer assisted tool is provided to assist the CDM Administrator in building the internal schema definitions used in the Test Bed.

12. Conceptual Schema Definition

A computer assisted tool is provided to assist the CDM Administrator in building the internal schema definitions used in the Test Bed. The items needed to be defined in a typical conceptual schema are shown on Figure C-7.

C.8 Migration Path

C.8.1 Hardware Migration Path

Figure C-8 shows a likely migration path for the Test Bed hardware. Figure C-8 does not imply a time scale, but portrays what is believed to be the most likely sequencing of hardware addition to the Test Bed.

It is realized that the Air Force's Integrated Applications Program Office is not in the computer hardware business. This hardware progression is shown to focus on the capabilities of the Test Bed as time goes on and to help the planning of the software activities required to support the evolution.

The items referenced in this section have been previously described (see Hardware Architecture). Items 1 & 2 are currently available.

Items 3 & 4 - The LAN configuration unit and self trouble shooting capabilities (Item 3) and the Direct LAN interfaces (Item 4) will be developed by the LAN vendors and will not impact significantly the ICAM software.

Item 5 - The workstations and their direct LAN interfaces will also be developed by independent hardware vendors. Some rehosting of Test Bed software and application programs is implied to take advantage of the work station concept.

Item 6 - The Message Handler processors will improve the throughput of the Test Bed. The processors are off the shelf computers (small mini, large micros) and are readily available. Software rehosting and possible extension (encryption/decryption) is implied.

Item 7 - The Back End processors are available (IDM, etc) as their introduction in the Test Bed will imply integration work with existing Test Bed software. The current NDML planned

EXTERNAL SCHEMA

USER ID AND TYPE (HUMAN OR SOFTWARE MODULE)

EXTERNAL SCHEMA ID AND AUTHORIZED USERS

SURROGATE ENTITY CLASSES W/I EXTERNAL SCHEMA

DATA ITEMS W/I SURROGATE

DATA ITEM PAIRS BETWEEN SURROGATES

SEC RELATIONSHIPS BETWEEN SURROGATES W/ LINK TO DATA ITEM PAIRS

DATA ITEM PAIRS

ALGORITHMS FOR DERIVED DATA ITEMS, CONSISTING OF:

- MODULE ID
- DATA ITEMS USED AS PARAMETERS
- SEC RELATIONSHIPS THAT SERVE AS "PATHS" FOR OBTAINING VALUES

Figure C-6. External Schema

CONCEPTUAL SCHEMA

ENTITY CLASSES
ATTRIBUTE CLASSES AND DOMAINS
ATTRIBUTE USE CLASSES - KEYS
- OWNED
- INHERITED
RELATION CLASSES - TYPE
- INDEPENDENT ENTITY CLASS
- DEPENDENT ENTITY CLASS
ALGORITHMS FOR DERIVED ATTRIBUTE CLASS

Figure C-7. Conceptual Schema

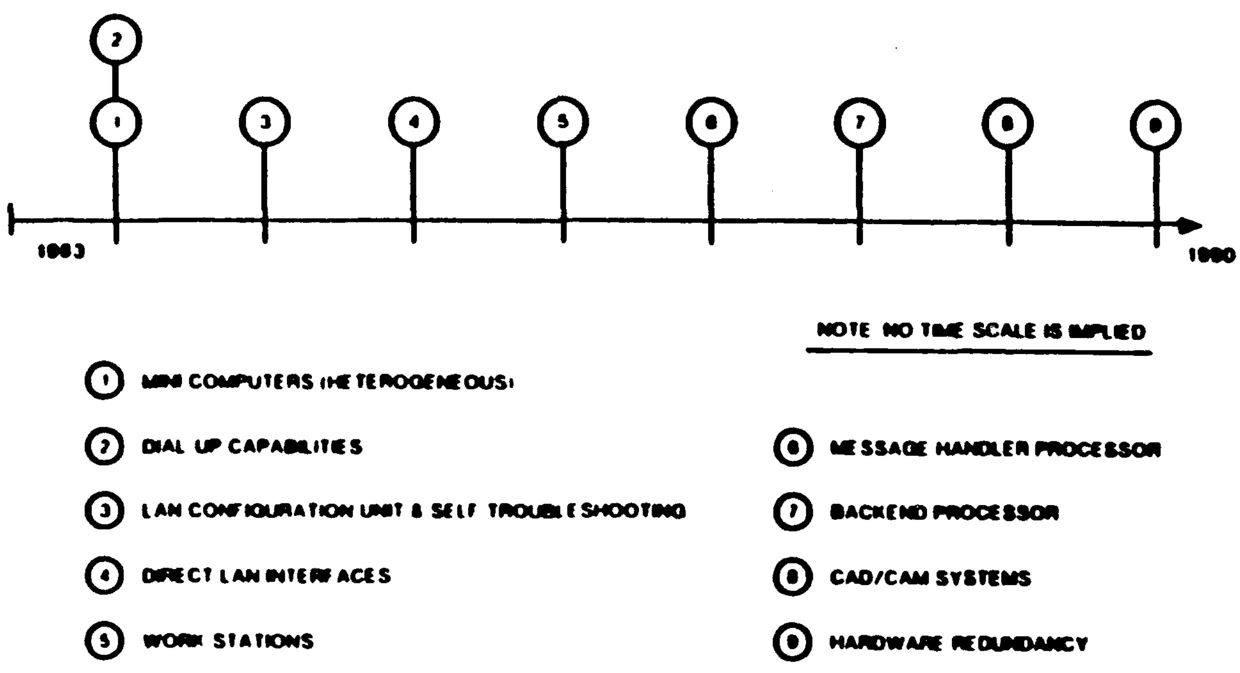

Figure C-8. Test Bed Hardware Migration Path for the Test Bed will facilitate this integration.

Item 8 - The CAD/CAM systems available at the time of this writing have limited or nonexistant networking capabilities. This development is unavoidable. The addition of CAD/CAM systems in the Test Bed will be facilitated by the IGES standardization work now in progress at General Electric and other research institutions. Significant software development may be required to interface the graphic databases to the Test Bed. Interfacing difficulties arise from the current lack of standardization among the graphic databases of various vendors.

Item 9 - Hardware Redundancy is available from several vendors (General Automation, etc.). Installation of redundant hardware will only be justified when the Test Bed moves into heavy production status.

C.8.2 Communication Software Migration Path

Figure C-9 shows the likely migration of the Test Bed Communication Software. Figure C-9 does not imply a time scale.

Item 1 - Binary File Transfer: Binary file Transfer capabilities are needed on the Test Bed to transfer graphic data, execute code between hosts and work stations, etc. The transfer of binary files will become practical when the direct LAN interfaces are provided.

Item 2 - Direct LAN Interfaces: The integration of Direct LAN interfaces into the Test Bed requires some modification to the COMM Software. Direct LAN Interface device drivers are to be integrated into the existing COMM Software.

Item 3 - Message Handling Processors: Rehosting some of the NTM functionality is required to take advantage of the Message handling Processors described in Figure C-9. The rehosting of the NTM functionality on the Message handling processor also requires software interface between the software resident on the Message Handling Processors and the software resident on Test Bed Hosts and back end data machines.

C.8.3 Distributed Database Management Migration Path

Figure C-10 shows the likely migration of the Test Bed Distributed Database Management Software. Figure C-10 does not imply a time scale.

1 BINARY FILE TRANSFERS

2 HIGH SPEED LAN INTERFACES - DRIVERS & SOFTWARE MODIFICATIONS

3 MESSAGE HANDLING PROCESSORS

Figure C-9. Communication Subsystem Migration Path

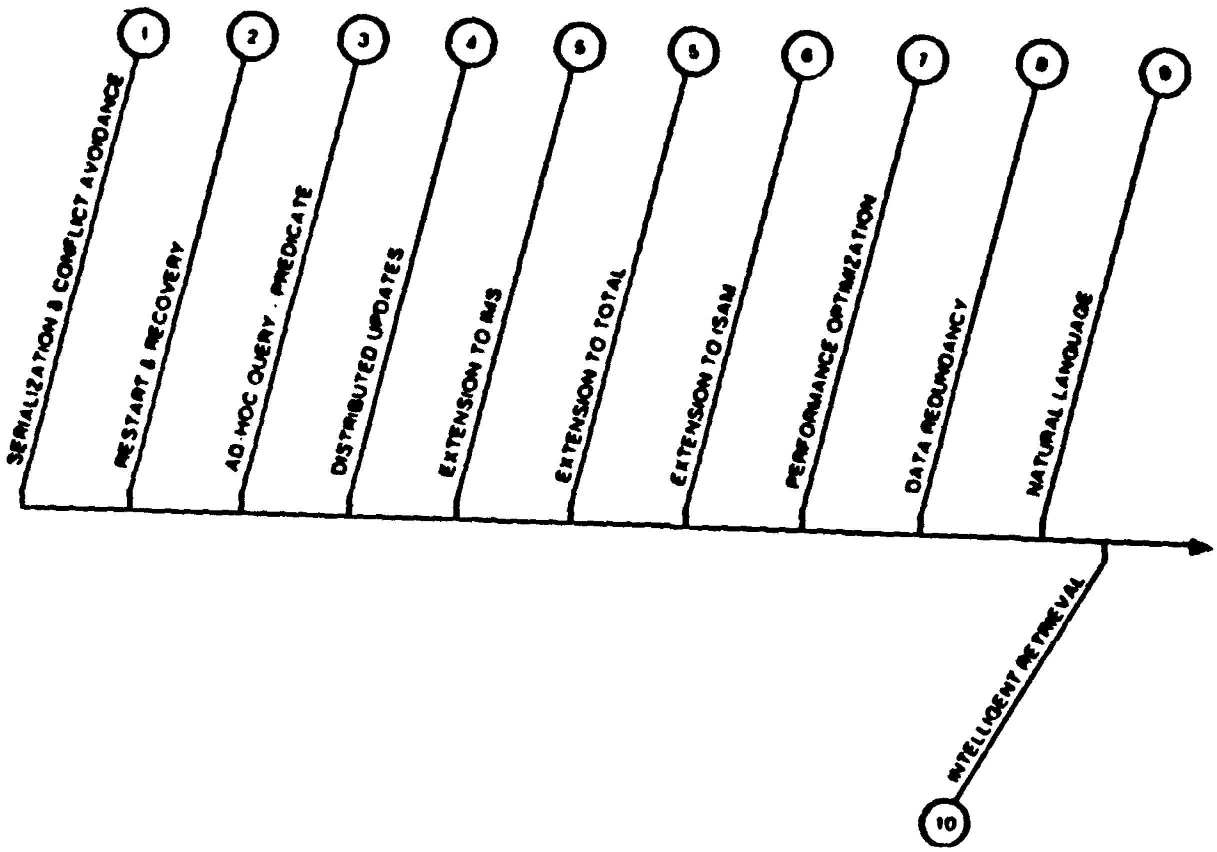

Figure C-10. Distributed Database Management Software

Item 1 - Serialization and Conflict Analysis [DPMARTIN]: Interference between two Application processes must be avoided if the distributed database management system is to yield consistent results. Interference between two Application processes may occur when both Applications attempt to update the same data or when an application process is reading data being updated by an other Application Process. The first case is the well documented interference problem occuring in distributed updates. The second case, perhaps less obvious, occurs in the Environment of the 620IM Test Bed. The following examples illustrate these two types of conflict.

1. Distributed Update Interference

Two application programs A and B update the same datum Xo. The following may occur:

a. No interference

A:	Xo	Xo + Xa	
B:	Xo + Xa	Xo + Xa + Xb	(final state)

b. Interferences

A:	Xo	Xo + Xa
B:	Xo	Xo + Xb

The final state of the database will depend on the actual sequence of execution of the two updates, and is in either case incorrect.

2. Distributed Read Interference
Two Application Programs A and B read the same datum Xo. Application Process A updates the datum, whereas B does not. The following may occur: B will either read Xo, or Xo + Xa. The value read by B depends upon the actual sequencing of A and B. If B uses the retrieved value to update an other database, consistency may be lost.

These two examples show that in the distributed environment, sequencing of the processing alters the end results. The following important practical implications result from the above examples:

a. Without system imposed sequencing (or serialization) consistency may be lost, without user warning

b. Recovery and normal processing are not necessarily equivalent

c. The system is not repeatable

In a centralized system, database locking is used to prevent the above mentioned problems. This approach, although feasible, results in severe degradation in system throughput and performance in the distributed environment [SDD-1]. Instead, systems such as SDD-1 rely on:

a. Time stamp driven processing

b. Application programs are characterized by the data they access. This characterization allows the determination of potential conflict, at run time, between application processes. The information required to characterized the application processes is determined at compile time or through analysis prior to execution time.

Item 2 - Restart & Recovery: Restart and Recovery capabilities must be provided to allow the restart and correction of a database which has been temporarily unavailable or which has been contaminated with erroneous data. The following scenarios are typical scenarios requiring a recovery action to be initiated:

a. Database is known to contain erroneous data.

b. Database contains inconsistent data.

c. Database has been updated with data which may be erroneous.

d. Database has crashed.

Hardware failures, system software and Application Process Software failures, operator errors are typical causes for the above scenarios.

The third scenario typically arises from updates derived from corrupted data contained in an other database. This failure mechanism makes the recovery of databases in a distributed environment that much more difficult to perform, since the propagation of an error is system usage dependent.

The restart/and recovery procedures implemented on the the Test Bed rely on the following concepts:

a. The states of the databases used as starting points in a recovery action must be valid and consistent.

b. Update Transactions are applied without omission or duplication.

c. Recovery processsing is equivalent to normal processing, so that the final states reached during normal processing and during recovery processing are identical.

d. Recovery procedures include the recovery of the databases and of the serialized transactions queues.

e. Recovery procedures are initiated manually. The operator can either roll back, roll forward, or reapply transaction streams. The initial and final states used in rolling back, rolling forward are specified by the user. Selected Transactions can be omitted from the input streams. This capability allows to repair databases contaminated by erroneous Application Processes by ignoring the Application Processes for the time being.

Item 3 - Ad-Hoc Query (Predicate): Ad Hoc query capabilities based on Relational Calculus enhance the power of the Test Bed System, by making test bed information more readily accessible to the user. This first level of ad-hoc query capabilities is not as user friendly as may be desirable. Friendlier query capabilities based on Natural language processing are described under Item 9. The Capabilities described here are viewed as the building blocks to the Natural Language Query Capabilities.

The Test Bed provides for predefined queries. The design of the 6201M and future project query processors provide the building blocks to the follow on ad-hoc capabilities.

The software required to input the query, parse the query, generate the query generators and distribute the COBOL query processor code must be developed. Of importance, is the software required to enforce authorized access to the data.

Item 4 - Distributed Updates: The software supporting distributed updates needs to be developed. Specifically, the distributed updates are performed in CBIS CLASS-I, that it is the updates are performed under control of the CDM. This capability brings about the full integration of the Test Bed databases. Reliable implementation of the distributed updates require services developed previously. The Guaranteed Delivery service developed under the 6201M contract ensured that Transactions were not lost nor duplicated. The serialization logic and conflict analysis logic developed under item-1 ensures maximum system throughput, repeatability and the abscence of dead locks.

Item 5 - Extension to IMS, To Total: The update and query capabilities developed against the 6201 specific instances of database managers are extended to other database management systems popular in the US Aerospace Industry. IMS and TOTAL have been singled out by the Integrated Applications Program Office to be of significant interest to the ISMC contractor. Extension to TOTAL, a network type database manager requires the development of a processor capable of translating Generic COBOL query processors into COBOL query processors written with TOTAL Data Manipulation Language. The extension of the update and query capabilities to IMS will be significantly more complex since IMS is a hierarchical database manager. The added complexity of the implementation of the Test Bed Neutral Data Manipulation Language on IMS stems from some undesirable properties of hierarchical database managers [CJDATE]. These properties result from the asymmetrical nature of hierarchies.

Get next operations must be qualified by an "under" clause to specify context, and additional information must be provided to solve problems introduced by the hierarchical data structures. In this context, many to many relationships are problematic. The hierarchical database has in addition the following, well documented, anomalies:

a. Insertion: It is not possible to insert a dependent record if the independent record has not been defined first.

b. Deletion: It is not possible to delete an independent record without deleting all dependent records associated with it.

c. Update: Hierarchical database managers contain redundant information. Changes to a dependant record may have to be duplicated, leading to a possible consistency problem.

Item 6 - Extension to ISAM [SATRE]: Indexed Sequential Access Method files (ISAM) are widely used in the US Aerospace industry [Program Office]. This fact provides the incentive to extend the implementation of the NDML to ISAM files. The difficulties arising from such an extension are as follows:

1. Lack of standardization

2. Access by any key other than the primary key (index) are clumpsy

3. Efficiency is function of the Activity in the database. None the less, it is believed that the NDML of the 6201M and 6203 Test Bed be implemented against ISAM files.

Item 7 - Performance Optimization: The 6201M Implementation of the Test Bed provided tactical optimization of the data queries. The access paths followed to retrieve & update data at the Conceptual and Internal levels are fixed. Data Aggregation is optimized upon the volume and location of the data retrieved. This optimization step is necessary but not sufficient. Further improvements in performance can be achieved by optimizing the access paths followed to retrieve or update data. Determination of the optimum path to retrieve data can be done via heuristics graph search techniques described in the literature of Artificial Intelligence [NILS]. Optimization of the access paths would be followed by the optimization of the query aggregation schedule, as done in the 6201 Test Bed.

Item 8 - Data Redundancy: The management of data redundancy is one of the most challenging problems of distributed data processing. Data redundancy has some benefits on system availability, and data query response time but has terrible effects on data integrity and system overhead in the update environment. The 6201M Test Bed did not address the issue of data redundancy. However, data redundancy must be dealt with by the current project since data redundancy may be imposed on the CDM Administrator by the very structure of the databases already in existence.

Furthermore, the systematic elimination of all redundancy may prove to be prohibitively expensive when considering the conversion cost of existing application programs. This, in addition, runs contrary to the very purpose of the Test Bed.

Data redundancy must be described in the CDM and rules must be derived for the selection of the location of target data under the query and update services environments. System ensuring the quick convergence of redundant copies of the data must be devised and implemented.

Item 9 - Natural Language Query: Natural Language Translators are coming of age, and are capable of bridging the gap between the English language and the formal - albeit user unfriendly - Relational Calculus. Natural Language Translators thus make the Test Bed Query capability available to a user not trained in Relational Calculus. The techniques used to construct Natural Language Processors have been described in the literature [PYGLO]. Natural Language Processors perform lexical analysis, syntax analysis, semantic and discourse analysis of the query. Ambiguities of the natural language are flagged and the Relational Calculus equivalent of the Natural Language is contructed. The Ad-hoc query capabilities described in Item-3 is a building block to the natural language based ad-hoc query capabilities.

Item 10 - Intelligent Retrieval From databases: Intelligent retrieval from a database involves deductive reasoning with the facts contained in the database. The understanding of the query may be performed through the Natural Language Processors described in Item-9 and may involve knowledge beyond that explicitly represented in the database. Common knowledge is typically omitted, and is however required to interpret the semantics of a query. The representation of the common knowledge is a state of research problem [NILS].

C.8.4 Test Bed Development Software Migration Path

The development of new Application Processes on the Test Bed is subject to the economic consideration set forth in the section entitled "Software Environment". Application Processes specifically developed for the Test Bed reflect the concern for accepted Software Engineering concepts of modularity, cohesion, minimum coupling, data, terminal and host independence while making maximum use of existing software or non procedural software to improve programming staff productivity. Figure MP-4 illustrates a likely migration path for the Test Bed Application Process Development Software. The nonprocedural data manipulation language, used for query, and for update purposes, has been implicitly described in the section "Distributed database Management Migration Path."

Item 1 - Fortran NDML PreProcessor: The 6201M contract developed a COBOL NDML Preprocessor. This preprocessor is used to prepass COBOL programs containing NDML statements. It is clear that a FORTRAN NDML Preprocessor must be developed promptly since a significant number of Application Processes are written in FORTRAN and since conversion to COBOL is either difficult or expensive. The functionality of the COBOL NDML pre processor is that of the pre processor written under contract 6201M.

Item 2 - Report Generator: Report Generators are an example of shared code boosting programmer's productivity. Many report generators exist today and are commercially available.

These systems, however, are not directly useable in the distributed database management context, and need to be modified or improved to provide a user friendly and economical way for the end user to display and format the information retrieved from the Test Bed.

Item 3 - Message Definition Syntax: Coordination and Communication in the Test Bed is performed via messages exhanged between the various cooperating application processes. Data and System Integrity is enhanced by checking messages which have predefined syntax. The Message Definition Syntax allows the definition of messages which can be easily checked by the Network Transaction Manager. The Syntax could be defined, for variable format messages, by transmitting the message format along with the message.

Item 4 - Distributed Macros: Further programmer productivity can be reaped by defining and implementing Macros for functions frequently invoked. In the context of the Test Bed, those Macros would perform distributed data retrievals and updates, and could be invoked by messages.

These Macros implement operations of particular importance to the manufacturing user. A Macro allowing the user tc issue a 1;2c

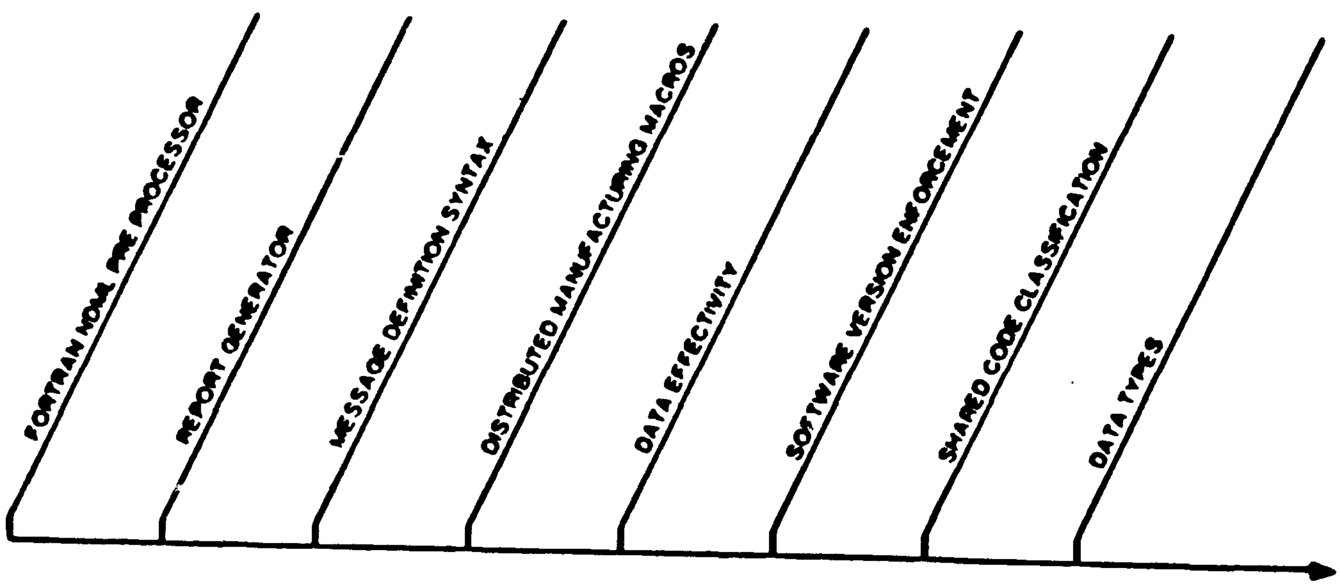

Figure C-11. Development Software Migration Path tool from the tool crib to a particular workstation is an example of such a Macro.

Item 5 - Data Effectivity: Data Effectivity is of particular interest to the manufacturing user. The implementation of data effectivity checks, supported by system software, would facilitate the use of effectivity data when it has been defined.

Item 6 - Software Version Enforcement: Application Processes and System Services are highly dependent upon the version of the CDM data used for their compilation. Enforcement of a match between Application Process version and corresponding CDM data version improves the reliability of the Test Bed by ensuring that Application Processes are using proper CDM data.

Item 7 - Shared Code: The Test Bed integrates many applications and it is used by many users who may not have the opportunity to communicate with one another. In such an environment the likelihood that users develop similar application processes is very high. To reduce the software development duplication, a method needs to be devised (Group Technology Classification) to allow for the systematic sharing of existing application processes among users.

Item 8 - Data Types: Abstract data types have a significant positive impact on software reliability. Abstract data types facilitate the manipulation of the data by defining allowable operations and by preventing the user from gaining access to the internal representation of the data. Capsules [SOFT] are defined to encapsulate both the internal representation and the operation to be performed on the data. The Capsules serve as non procedural building blocks to the Application Programmers, and further improve productivity by relieving the Application Programmer from knowing the details of the internal data structure and associated constraints. System services must prevent direct access to the data for data types to be of maximum utility.

C.8.5 Graphics

The need for retrieving and for transmitting graphic data files will arise as the Test Bed moves into the manufacturing environment. Present and future graphic CAD/CAM systems are turn key systems, which will be progressively brought under the Test Bed. The following developments will enhance the use of graphic data on the Test Bed.

1. Virtual Terminal Graphic Capabilities

   Line graphic capabilities on the Virtural Terminal will facilitate the display (but not editing) of graphic data prepared on the turn key graphic systems. This capability allows the display of graphic data in the shop without requiring special hardware, as well as the display of line graphics generated by other Test Bed application processes.

2. Integration of Graphic Databases

   The graphic databases of turn key systems will be progressively integrated as documentation of existing graphic database structures and direct LAN interfaces become available. Conversion to a neutral format (IGES) will further facilitate the integration of graphic data.

C.8.6 Control Architecture Migration Path

The Control Architecture Migration Path shown on Figure C-12 facilitates the implementation of new application processes on the Test Bed by providing more elaborate methodology and tools as time progresses.

1. Test Bed Administrator Guide

   A guide is provided to the Test Bed administrator to assist him or her in the task of defining:

   a. Users who should have access to the Test Bed

   b. What information should be provided to these users

   c. Which application processes are required

   d. Which screens, etc. are required

   e. How to audit system and data usage made by the users

   f. How to use the Test Bed Data Dictionary

2. Programmer's Guide

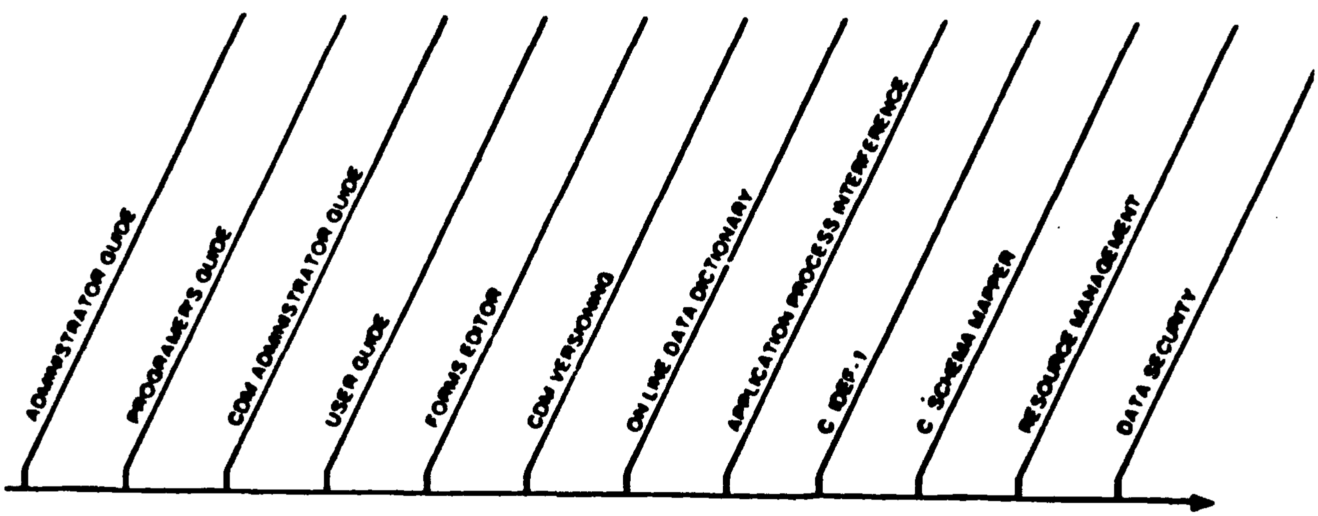

Figure C-12. Control Architecture Migration Path

A guide is provided to the application programmer to assist in the design and coding of application programs on the Test Bed. The programmer's guide describes:

a. Neutral Data Definition Language

b. Neutral Data Manipulation Language

c. Services provided by the Test Bed

d. User Interface services

e. Prepassing and compilation procedures

f. Test Bed operating principles

g. Test Bed error messages

3. CDM Administrator Guide

The CDM Administrator Guide assists the CDM Administrator in creating the schemas (external, internal, conceptual) required to integrate new applications and databases on the Test Bed. The CDM Administrator guide describes:

a. IDEF1 methodology and its extension

b. How to create an Internal schema for an existing database

c. How to create and augment the Conceptual schema

d. How to create an External schema

e. How to define the mappings between the schemas

f. How to store, edit, schemas on the CDM

g. How to use the CDM data dictionary

4. User's Guide

The user guide describes the operations of the Test Bed to the end user. This guide describes:

a. The User Interface services, from the end user point of view

b. Error messages

c. High level explanation of the Test Bed operating principles

5. Forms Editor

The forms editor is a 6201M shortfall. This total should be developed rapidly to give the user the flexibility required to define forms graphically (by moving strings on the CRT) and textually.

6. CDM Versioning

Versioning of the CDM is a 6201M shortfall. This important service must be provided, since it will allow making changes to the CDM without having to preprocess and recompile all application processes. The granularity of CDM Version numbering system must allow to single out those application processes affected by a change to the CDM data. It is noted that some changes (passwords for example) have no impact on the recompilation of application processes.

7. On Line Data Dictionary

An on line data dictionary is needed to assist the CDM Administrator in the burdensome task of keeping track of data definition, and semantics. The data dictionary is updated automatically when changes are made to the CDM.

8. Application Process Interference

Application process classes need to be defined automatically by the Test Bed. An application process class may contain one or more application processes, provided that they are using the same logical read set and write set of data (SDD-1). The grouping of application processes into classes allow significant improvements in system performance. This assignment must be error free since conflict and inconsistencies may result from the interleaved processing of application processes of the same class. Automation of application process class definition is thought to be more effective and reliable than manual definition.

9. Computer Assisted IDEF1

The methodology developed for the integration of existing databases calls for the development of normalized IDEF1 data models (DACOM). A computer assisted IDEF1 model building tool will make this process more productive, less error prone. It will provide the hand holding required for transitioning the Test Bed away from its developers. The IDEF1 model builder cannot be fully automated, but it can be made interactive and exhaustive. This tool should accept input from the model builder, from IDEF1 models already stored in the CDM or from the schemas of existing databases already defined.

10. Computer Assisted Schema Mapper

The task of mapping schemas is thought to be tedious and is a good candidate for computer assistance. Schema mapping cannot be automated, for it requires too much common knowledge about data semantics. Schema mapping can, however, be made highly interactive and the computer be of significant assistance in checking completeness and self consistency of the mappings.

11. Resource Management

The resource management capabilities provided for the 6201M Test Bed are satisfactory for the laboratory or experimental phase of the life of the Test Bed. However, as the Test Bed expands to full production status, improved resource management capabilities must be provided for budgetary control.

12. Data Security

The ad-hoc environment poses added data security problems. In this environment, data security cannot be enforced via access control. More granular data security information must be provided, to allow distinguishing between valid and invalid data usages of a legitimate user. Subdividing users into security levels is a convenient way of enforcing direct data security among legitimate users of a database. Enforcement of this additional security constraint must be provided in the Test Bed via shared system services.

The need for data encryption may arise as more sensitive data (commercial or military) is brought under the Test Bed. Approved encryption algorithms (DES) may be used on the Test Bed, and be implemented on message handler processors and workstations.

C.9 Prioritized List of Development Activities

The following is a prioritized list of development activities required to transition the Test Bed from its state of completion at the end of the 6201M contract to full production status. The development activities called are those described in Section 8.

PHASE 1: DEMONSTRATION OF BASIC CAPABILITIES

1. Full Query Capabilities:

    a. Serialization and conflict analysis
        1. Time stamp processing or other conflict avoiding technique
        2. Transaction class processing
    b. Restart/Recovery
    c. Binary File Transfer

    d. Fortran NDML Pre processor

2. Ad-Hoc Query Capabilities

    a. Ad-Hoc Query (PREDICATE)

3. Distributed Update Capabilities

    a. Distributed Updates

PHASE 2: DEVELOP, AUTOMATE INTEGRATION METHODOLOGY

a. Data Redundancy
b. Computer assisted IDEF-1
c. Computer assisted schema mapper
d. CDM Versioning
e. CDM Utilities
f. On line Data Dictionary
g. Test Bed Administrator Guide
h. Programmer's Guide
i. CDM Administrator Guide
j. User's Guide

PHASE 3: PREPARE FOR MIGRATION TO MANUFACTURING ENVIRONMENT

a. Extension to IMS and TOTAL
b. Extension to ISAM
c. Data effectivity
d. Application process interference classification
e. Performance optimization
    1. Retrieval optimization - conceptual level
    2. Retrieval optimization - internal level
f. Virtual terminal line graphic capabilities
g. Resource management
h. Data security
i. Workstations
j. LAN configuration unit & self troubleshooting

PHASE 4: DEVELOP TOOLS TO IMPLEMENT NEW APPLICATIONS

a. Forms editor
b. Report generator
c. Message definition syntax
d. Distributed manufacturing macros

PHASE 5: IMPROVE TEST BED PERFORMANCE

a. Direct LAN Interfaces
    1. Procure, install hardware
    2. Modify COMM software
b. Message handler processors
    1. Procure, install hardware
    2. Modify NTM software
c. Backend processors
    1. Procure, install hardware

2. Modify query generators

d. Hardware redundancy

PHASE 6: GRAPHICS CAPABILITIES

a. Graphic/LAN direct interfaces
b. Integration of graphic databases

PHASE 7: ADVANCED CAPABILITIES

a. Shared code classification
b. Abstract data types
c. Natural Language Query Processor
d. Intelligent database retrieval

C.10 Bibliography

(DP MARTIN) J. Martin, Design and Strategy for Distributed Data Processing, Prentice Hall, 1981.

(A TOFFLER) A. Toffler, The Third Wave, Bantam Books, 1980.

(D ELLIOTT) "One Chip Carries Out Ethernet Protocol," Electronic Design, September 30, 1982.

(WEITZ) Wietzman, Distributed Micro/Minicomputer Systems, Prentice Hall, 1980.

(AD MARTIN) J. Martin, Application Development Without Programmers, Prentice Hall, 1982.

(NILS) Nils J. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto, CA, 1980.

(C DATE) C.J. Date, An Introduction to Database Systems, Third Edition, Addison Wesley, 1982.

(DACOM) M.E. Loomis,"Integration Methodology," Design and Status Review, November 1982. D. Appleton Company, Manhattan Beach.

(CODASYL) A Framework for Distributed Database Systems: Distribution Alternatives and Generic Architectures, Codasyl Systems Committee

(SDD-1) PA Bernstein, JB Rothnie et al., The Concurrency Control Mechanism of SDD-1, A System for Distributed Databases; Technical Report No. CCA-77-09. Computer Corporation of America.

(SATRE) S. Atre, "Database Structured Techniques for Design, Performance and Management," Wiley Interscience, 1980.

(PYGIO) Paul Y. Gloess, "Understanding Artificial Intelligence," Alfred Publishing Co., 1981.

(SOFT) H. Freeman, PLEWIS, Software Engineering, Academic Press, 1980.

D. Appleton, Information Resource Management, DACOM, 20 October 1982.

R. Jones, IISS Enhancements Document, Working Paper, SofTech, November 1982.

U.S. Government Printing Office 1992—648-127/62285