User Centered Systems Analysis of the North Dakota Intermodal Management System

Douglas E. Benson

UGPTI Publication No. 114
February 1997

ABSTRACT

This paper presents the results of a user centered systems analysis of a proposed software system, the North Dakota Intermodal Management System. The study surveyed and evaluated several object-oriented analysis methods to select the one that best supports user centered development. This examination determined that Jacobson's Use Case Driven Approach best meets that criterion.

The study continued the analysis with an initial application of Jacobson's methodology to the software. This part of the analysis included the identification of high level use cases and the object specification of an example use case. In doing so, the paper augmented Jacobson's methodology by introducing a priority ranking of the primary actors in the software system and by applying user centered design principles and guidelines to the development of a system prototype.

The paper concluded by presenting an analysis of alternative software development environments for implementing the software. These alternatives were evaluated and presented to the agency considering system implementation. The analysis examined the potential functionality for each alternative and compared each alternative with the other possible software development environments.

ACKNOWLEDGMENTS

The author acknowledges the assistance of Dr. Denver Tolliver of the Upper Great Plains Transportation Institute and Norlyn Schmidt of the North Dakota Department of Transportation. These transportation professionals provided invaluable guidance and insight into the development of a North Dakota Intermodal Management System. Additionally, the author would like to acknowledge the contributions made to this paper by Dr. Kendall Nygard of the Computer Science Department at North Dakota State University.

Dr. Nygard, along with Dr. Paul Juell, Dr. D. Bruce Erickson, and Dr. Don Andersen of North Dakota State University, made significant contributions to the computer science aspects of the study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Introduction to the Paper

The work described in this paper concerns steps in the design and development of software for transportation analysis and decision support. The software is known as the Intermodal Management System (IMS), and it is essential that the software design of the IMS be user centered. Within that context, the study had three primary objectives (Figure 1). The first was to survey and evaluate alternative object-oriented analysis and design methodologies to select the one that best supports the user centered design approach. The outcome of this effort was to select the use case approach pioneered by Ivar Jacobson (Jacobson, 1992). The second objective was to initiate the actual application of the use case approach to the IMS, developing high level use cases and object classes for a significant portion of the IMS. This exercise characterized the applicability of the approach and illustrated many of the major issues that must be addressed in implementing the entire system. Moreover, the paper seeks to strengthen the application process by applying user centered design principles to an IMS prototype design. The third objective of this paper was to survey alternative software development systems for possible IMS implementation. This effort was done in response to the potential users of the system.

The paper begins with an introduction to the IMS. It is first explained from a national legislative perspective and then from North Dakota's perspective (UGPTI, 1995) by outlining the IMS objectives delineated by the North Dakota Department of Transportation (NDDOT), the government entity considering IMS development. This is followed by a description of the selection of object-oriented technology for this analysis.

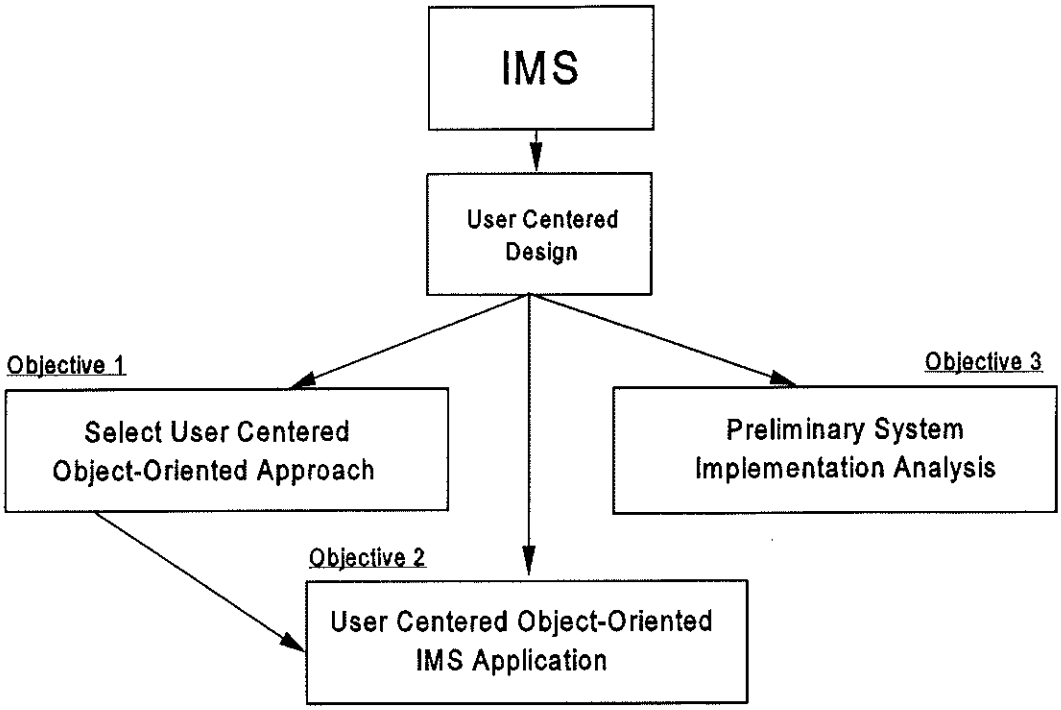


Figure 1. The Objectives and Process of the Study.

The literature review focused on selecting a user centered, object-oriented methodology. The format of the literature review is introduced first, followed by an overview of object-oriented technology. The chapter continues with a survey and an examination of several major object-oriented methods and concludes by selecting a user centered object-oriented methodology, Jacobson's use case model.

Chapter 3 initiates the application of Jacobson's use case model to the IMS. The chapter outlines the application of the use case model at a high system level and also, through the use of an example, in greater depth. A comparison with another analysis and design approach is used to highlight the advantages the user centered, object-oriented

methodology offers. The chapter introduces the topic of interface design, an important issue in user centered design.

Chapter 4 reviews and examines user centered design principles, applying them to the development of an IMS prototype. The IMS prototype is part of the analysis and design process utilized to elicit user feedback early in the process, and the use of user centered design principles in the development of the IMS prototype strengthens its part of the process. User centered design principles are presented from several viewpoints, and a computer use survey is utilized to provide basic user information for applying these principles. Design decisions for the IMS prototype are outlined, and the chapter concludes with a discussion of a user centered IMS prototype.

Chapter 5 surveys alternative software development environments as requested and reported to the NDDOT (UGPTI, 1995). This analysis was developed to present software development alternatives to NDDOT officials and is presented as part of the user centered design process.

Chapter 6 presents the conclusions. These detail the study's efforts in accomplishing the objectives of the paper, the selection of a user centered, object-oriented methodology, the initial application of the methodology to the IMS, and the presentation of alternative software development systems for possible IMS implementation.

1.2 Introduction to the Intermodal Management System

The Intermodal Management Software System (IMS) is a computerized management system being considered for development by the NDDOT within the framework provided by the Intermodal Surface Transportation Efficiency Act (ISTEA) of

1991. The ISTEA directed the states to monitor and enhance the performance of intermodal transportation within their states and proposed a management system for intermodal facilities and systems that provides for the integration and improvement of all of their transportation systems. This requirement was relaxed in 1995. The Federal Highway Administration (FHWA) and the Federal Transit Administration (FTA) define an intermodal system as a transportation network for moving people and goods using various combinations of transportation modes. The IMS will be a transportation management system designed to meet the expectations of ISTEA by providing a computer-based system to assist in the administration and evaluation of the state's intermodal transportation system.

The Federal Register, Vol. 58, No. 39, contains the proposed rules detailing the minimum requirements and objectives of an IMS. As described in the Federal Register, they are

a. Identification of intermodal facilities.

b. Identification of efficiency measures and performance standards.

c. Data collection and system monitoring.

d. System and facility performance evaluation.

e. Strategy and action identification and evaluation.

f. Implementation.

1.3 Objectives of the North Dakota IMS

The NDDOT envisions a number of objectives for the state's IMS (UGPTI, 1995). A major goal of the system is to provide a mechanism for optimizing state highway investment analysis to maximize the use of state highway funding. To reach that goal, the

collection and analysis of railroad freight data and motor carrier freight operations are included in the IMS. The traffic shifts among different modes of transportation also must be monitored to measure the potential impacts modal shifts have on investment decisions. To meet that objective, the IMS will provide functionality to easily assess modal shifts. Among the other primary goals representative of those outlined by NDDOT (UGPTI, 1995) are 1) provide information regarding the volume handled and traffic generated from major freight traffic facilities; 2) provide facility-specific and commodity-specific traffic data for major traffic generators which will support improved Average Daily Traffic and Equivalent Single Axle Loads measures; 3) provide information outlining access routes to and from airports, grain subterminals, and other major facilities; and 4) provide truck terminal information useful in analyzing access for large trucks to and from the National Highway System.

A major operational goal of the IMS is to integrate it with existing management systems (UGPTI, 1995). These management systems are the traffic monitoring system and the pavement, bridge, safety, and public transportation management systems. Integration and coordination with these systems will complement the activities of all systems. The NDDOT also wants the IMS to be as practical as possible and targeted toward the major intermodal issues.

1.4 Preliminary Assessment of the IMS

A preliminary assessment of the IMS revealed that the system has the potential to be large and complex. An evaluation of other states' efforts showed that significant projects were in development. A presentation of California's plan outlined an extensive and elaborate series of management systems and databases being incorporated into their IMS (Carter,

1993). A New Mexico IMS planning document identified several major projects as part of the IMS development (Barton-Aschman, 1993).

The North Dakota IMS has a significant number of databases, which are listed in Table 1, and will be accessing at least one other management system, the Pavement Management System (PMS). Potentially, the North Dakota IMS could be interacting with a number of other systems including a GIS and several other transportation management systems. The background surveys, in Appendix A and B, report many uses and functions for the system. In all, the North Dakota IMS will be a large, complex system when it reaches its full potential.

Table 1. The North Dakota Intermodal Management System Databases.

IMS Database	Database Description
Coal	Collection of Coal Handling Facilities
Airports	Attributes of North Dakota Airports
Transit	Database of Transit Facilities
Fertilizer	Record of Fertilizer Distribution Centers
Grain Elevators	Characteristics of Grain Elevators
Motor Carrier	Major Motor Carrier Terminal Information
Sugar Beet	Sugar Beet Collection/Processing Centers

Object-oriented technology has been recommended by many software methodologists as a good technology for developing large and complex systems (Booch,

1991; Graham, 1994). This paper considered those recommendations for analysis and examined object-oriented technology within the user centered approach to development.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction

The objective of the literature review was to select a user centered, object-oriented analysis and design methodology, one of the primary objectives of the study. To accomplish that task, the literature review opens with an overview of object-oriented programming to introduce and trace the evolution of object-oriented technology. The literature review surveys and examines several major, object-oriented analysis and design methodologies, including those of historical note and concludes with the selection of a user centered, object-oriented analysis and design methodology.

2.2 Object-oriented Programming Overview

Object-oriented programming is a programming methodology based upon the representation of real-world entities as objects in a computer program (Entsminger, 1990). Objects in an object-oriented programming system incorporate, or encapsulate, the attributes and activities of the actors in a real-world system that the user is seeking to model. Real-world systems consist of actors having characteristics and behaviors interacting in the system corresponding to the objects in an object-oriented system. The objects in object-oriented programming form the building blocks of a system allowing the program to more closely emulate the real-world problem domain. This is contrasted with traditional structured programming which separates the characteristics and behaviors of a concept and delineates a more static control flow than object-oriented programming.

2.2.1 Object-oriented Concepts

The basic concept of object-oriented programming is that a collection of data and the operations that are normally performed on that data are very closely related and should be treated as a single entity rather than as separate things (Peterson, 1987). Several concepts are key to developing an understanding of object-oriented programming These key concepts are

Objects: Objects are the basic structure of object-oriented programming. An object encapsulates data and the data operations into one construct. It models a real world actor in a system in the sense that it contains the characteristics (data) and the behaviors (data operations) of an actor or entity.

Data Abstraction: An abstraction represents the essential characteristics of an object or concept without considering the underlying details. Data abstraction is a form of abstraction where the details of the underlying algorithms are hidden (i.e., they are an abstraction), and the type of data which those algorithms manipulate is also a high-level concept, i.e., an abstraction (Berard, 1993). This allows a designer to define an object by its attributes and operations apart from considering the implementation details. Data abstraction allows objects to be treated as "black boxes" without consideration of the underlying implementation of the objects (Berard, 1993).

Encapsulation: Encapsulation encloses the operations and the implementation of the data structures inside an object, thereby hiding it. The designer sees the object in terms of its attributes and operations, not in terms of the implementation details.

With encapsulation, we can tell the object what to do, but the details of how it works have been enclosed inside the object.

Polymorphism: Polymorphism is the capacity of a single entity (i.e., a message or an operator) to have different, but proper, interpretations across various different types of objects. The same method may be used regardless of the type (or form) of the objects upon which it is used. With polymorphism, many different types of objects can receive the same message, but correctly react to it in their own way (Berard, 1993).

Inheritance: Inheritance is a mechanism whereby objects acquire and/or inherit the characteristics and behaviors of another object, a parent object. Objects inherit all the attributes and characteristics of the parent object, and a hierarchy of objects can be created from inheritance. With inheritance, all changes made to the parent object propagate throughout the system to the child objects.

Classes: A class is an entity which is used to create instances of that type of object. A class is the template or blueprint for creating a category of objects (Booch, 1991). Among other things, a class describes the interface these items present to the outside world.

Message Passing: Objects communicate by passing messages to one another.

2.2.2 Object-oriented Background

Object-oriented programming started with the development of the Simula language in Norway in the late 1960s (Dahl and Nygaard, 1966; Berard, 1993). Table 2 provides an outline of the evolution of object-oriented technology, starting with Simula. Simula was

developed to provide a comprehensive programming language that was favorable for programming discrete event simulations. Discrete event simulations do not lend themselves easily to programming in traditional procedural languages where the control of program flow is functionally based. Simulation is modeled more effectively with an approach to programming where program control flow is based upon entities or objects changing state through the influence of other actors or objects in the system. Simula allowed program designers to more directly reflect the natural structure of a simulation problem, a system of objects, and interactions among the objects.

Smalltalk, an object-oriented programming language developed at the Xerox Palo Alto Research Center for use on an experimental small personal computer called the Dynabook, was introduced in the 1970s (Kay, 1977; Peterson, 1987). Kay (1977) describes Smalltalk as a language allowing the programmer to deal with objects and messages among objects to create "activities" that provide the framework of a program. Smalltalk became the "object-oriented" language used on the Dynabook and used a windowing scheme to program the computer.

The use of object-oriented concepts in Artificial Intelligence (AI) also began in the 1970s. The use of knowledge representations in AI using semantic networks and frames were influenced by the object-oriented concepts of objects and classes. In turn, the complex notions of inheritance in AI contributed to the evolution of inheritance in object-oriented programming (Graham, 1994).

Table 2. The Evolution of Object-oriented Technology.

Time Frame	Developments
Late 1960s	First Object-Oriented Language, Simula.
1970s	Introduction of Smalltalk and windowing scheme. Object-oriented concepts appear in AI. AI contributes to the evolution of inheritance.
1980s	Extensive UI development influenced by Smalltalk. Introduction of Ada, C++, and Object Pascal. Object-Oriented Design, Object-Oriented Analysis appear. Introduction of Object-Oriented Database Management Systems. Object Management Group (OMG) formed.
1990s	Emphasis on standards with many analysis and design methods. Booch, Rumbaugh, Coad and Yourdon among major methods. Ivar Jacobson's Use Case Model introduced. Unified Method expected 1996.

The 1980s brought extensive development in the User Interface (UI) area, and Smalltalk's ideas and windowing scheme were important to these developments (Graham, 1994). The window interface, with its collection of objects on the screen, was particularly well-suited for development with an object-oriented approach. Several conventional languages, such as C and Pascal, were extended to incorporate object-oriented concepts and led to the introduction of the widespread use of the object-oriented languages, C++ and Object Pascal.

Object-oriented Design (OOD), Object-oriented Analysis (OORA), Object-oriented Domain Analysis (OODA), and Object-oriented DataBase Management Systems (OODBMS) evolved in the 1980s (Berard,1993). The earlier work in object-oriented technology had not seriously considered design issues (Berard, 1993). Grady Booch was among the first to formalize a design approach drawing upon the work of Russell J. Abbott, an approach using nouns and verbs of textual language. Booch was looking for a mechanism to incorporate software engineering into the teaching of Ada and referred to his approach as object-oriented design (Booch, 1982). Booch and Edward Berard both incorporated object-oriented design exercises into their Ada instruction during the 1980s. In 1986, Booch published a paper outlining a method for using some of Ada's features in an object-oriented manner. Many others began developing approaches to object-oriented design of which two of the more prominent, Coad/Yourdon and James Rumbaugh's OMT, were introduced in 1991 (Coad and Yourdon, 1991; Rumbaugh et al., 1991).

Concern has shifted to standards (Graham, 1994; Berard, 1993). The Object Management Group (OMG), formed in 1989 with such companies as Hewlett Packard, AT&T, SUN, and others, has sought to develop standards for object technology along with an awareness of object technology and open systems. This shift reflects a change in emphasis from programming to design and analysis as exhibited by the proliferation of design and analysis methods published by the early 1990s. By 1995, two of the leading methodologies, Booch and OMT, were attempting to merge the two technologies into a method labeled the Unified Method (Booch and Rumbaugh, 1995). By late 1995, a draft version of the Unified Method was available and a final version scheduled for the middle of

1996. With a major share of the commercial market and utilizing concepts from several influential thinkers in the field, including Ivar Jackson, developer of the Use Case described in Section 2.3, the Unified Method appears positioned to be the major force in object-oriented design and analysis in the late 1990s.

2.3 Object-oriented Design and Analysis Methods

This section describes several of the most influential of the object-oriented design and analysis methods. The discussion of these major methodologies focuses on presenting a description of their major concepts and what each has to contribute to object-oriented technology. This discussion is presented within the framework of selecting a user centered, object-oriented analysis and design methodology. This section includes methods that are generally inclusive of both object-oriented analysis and object-oriented design, although it is important to note the difference between the two processes. Object-oriented analysis has a higher level of abstraction than object-oriented design and is the decomposition of problems or systems into their component parts. The process of specification of user requirements and system structure and function independent of the implementation of the system is object-oriented analysis. The object-oriented analyst considers and comes to an understanding of the system in terms of objects, object behavior, and system behavior. In comparison, object-oriented design is less abstract than analysis, but more abstract than implementing the system solution in program code. Graham (1994) states that object-oriented design methods share the following basic design steps:

1. Identify objects and their attribute and method names.
2. Establish the visibility of each object in relation to other objects.

3. Establish the interface of each object and exception handling.

4. Implement and test the objects.

The discussion begins with a look at an important, early contribution by Russell Abbott (1983).

2.3.1 Abbott (1983)

Abbott (1983) presents the use of natural language to extract the objects and methods of a system from the textual description of a problem. The nouns in the description become the objects in the system while the verbs become the methods, behaviors, or classification schemes associated with the objects. Nouns can be broken down into proper and improper nouns and verbs into doing, being, and having. While not a formal method, it is presented here briefly because of its use in subsequent methods. Many of the early methods use Abbott's textual analysis.

2.3.2 Booch (1986)

This is the oldest of the object-oriented methods and was developed mainly as a method for using the Ada programming language in an object-oriented style (Graham, 1994). Booch introduced the concept of "object-oriented design" to the Ada community (Berard, 1993). The approach is based on Ada's information hiding, but does not include important object-oriented concepts such as inheritance and polymorphism. The method constructs data flow diagrams (DFD) of the system and identifies the bubbles and data stores in the DFD as objects in the problem space. The system methods are derived from the process bubbles in the DFD. This methodology presents Booch's early ideas for

object-oriented design. Booch's later and more complete design and analysis approaches are described in Section 2.3.7.

2.3.3 General Object-Oriented Design (GOOD) (1986)

This method is similar to Booch's 1986 approach and uses layered data flow diagrams to identify objects in the system. Analyses of the entities in the system becomes the objects; the transformations of the entities in the system becomes the methods. Classes of objects are discovered by examining the flow of data and control. Like Booch, it uses a top-down seniority hierarchy on the objects and is closely linked to systems development using the Ada language.

2.3.4 Hierarchical Object-Oriented Design (HOOD) (1989)

HOOD is a method similar to GOOD and was directly influenced by it (Graham, 1994). This method has two kinds of objects, passive and active (HOOD, 1989). Passive objects may only use the services of other passive objects, whereas active objects may use the services of any object. Objects at the highest level of abstraction are decomposed into other objects in a top-down manner, with further decomposition of any resultant objects.

HOOD has, in effect, two hierarchies: a compositional one and a usage one. The usage one is the network of the passive/active relationships and is similar to a network schema. The compositional hierarchy details the top-down decomposition of objects.

The basic design step is a conventional diagramming effort after an Abbott textual analysis. The diagramming may include context diagrams giving hardware objects and external interfaces, DFDs displaying abstract data types and data pools, and state transition diagrams (STD) giving active objects and object-based control structures. The next step is

to produce an informal solution strategy using a natural language outline and HOOD diagrams describing the current level of abstraction. Last, formal HOOD diagrams are developed displaying the parent-child relationships and operations (composition hierarchy), usage hierarchies, implementation links, exceptions, and the data flows.

HOOD is based on the Ada language and, as such, is not a complete object-oriented method. It does not have inheritance or polymorphism, major object-oriented concepts, and has little support for reusability, a major object-oriented benefit.

2.3.5 Object-Oriented Analysis and Design (OOA/OOD) (Coad and Yourdon, 1991)

OOA/OOD was the first widely published account of a reasonably complete object-oriented analysis method (Graham, 1994). An important goal of the method is the reduction of a problem's complexity and the system's responsibility within it. The method enumerates eight principles, including abstraction, encapsulation, and inheritance, for the management of complexity within the problem space. It shifted emphasis from design to analysis with no major difference between analysis and design graphical notations. It also removed language constructs, such as Ada data structures, from the methodology making it more widely applicable.

Two key object-oriented concepts as defined by this method are

Object: An abstraction of an entity about which information has to be kept; an encapsulation of attribute values and their exclusive services.

Class: A description of one or more objects with a uniform set of attributes and services including the process of how to create new class objects.

Their analysis consists of five stages:

Subject Layer: The problem is decomposed into subjects which may be thought of as subsystems or class categories. Subjects are a mechanism for partitioning large, complex models and for organizing the subsystems within a problem. Subjects should contain approximately five to nine objects. This process can take place at various stages of the analysis and can be used for an initial decomposition or to organize the model after objects have been identified or refined (Graham, 1994).

Object Layer: Objects are identified in greater detail by locating the system entities that perform activities.

Structure Layer: Two types of structures must be identified: classification structures and composition structures. The classification structures incorporate inheritance.

Attribute Layer: Attributes are detailed, and extended relational analysis (ERA) is used to develop modality and multiplicity relationships.

Service Layer: A service is an object's specific behavior. The service layer specifies each object type's methods for creating and deleting instances, getting and putting values, and more individualistic object behavior. This layer captures methods and message connections between class and objects and defines the functional aspects of the system.

These layers are developed using five activities outlined by Coad and Yourdon: 1) Finding Class & Objects, 2) Identifying Structures, 3) Identifying Subjects, 4) Defining Attributes, and 5) Defining Services.

The analysis (OOA) section of the methodology now transitions into the design (OOD) section. Coad and Yourdon (1991a) explain how OOA and OOD relate to each other. "The OOA layers model *the problem domain and the system's responsibilities.* The OOD expansion of the OOA layers model a particular *implementation*" (page 178).

The object-oriented design adds four components to the OOA layers making the overall process more specific to design issues and lower level concerns. The OOA results are refined into four components: the problem domain component, the human interaction component, the task management component, and the data management component.

In the problem domain component, objects and classes that can be reused from previous projects are identified, and classes are organized into groups. Programming language specifics are addressed in terms of making changes to accommodate inheritance structures. Additional activities in this component include addressing performance and data storage considerations.

The human interaction component identifies the users of the system who are defined along with their characteristics and work task scenarios. A command hierarchy is developed and refined by organizing this hierarchy using common human-computer interaction guidelines. Prototyping is used for testing human interaction.

The task management component identifies the tasks in the system including event-driven and clock-driven tasks. Also defined are those tasks that have priority in the system and those that are critical to the system. The number of tasks should be minimized to reduce complexity. After task identification, the tasks can be specified by defining what the task is, how the task coordinates within the system, and how the task communicates.

The first activity in the data management component is to determine what type of data management process or system will be used, such as flat files or database technologies. The proper tools for managing the data within the data structures must be chosen as well as designing a data layout. The corresponding services to actuate and process the data within the objects and the system are designed. At this point, the OOA/OOD model, a multi layer, multi component model is complete.

2.3.6 OMT (1991)

OMT (Rumbaugh et al., 1991) is widely regarded as one of the most complete object-oriented analysis methods (Graham, 1994). OMT describes object-oriented as a way to develop and organize software that collects objects incorporating behavior and data structures. OMT has three phases of analysis and design: the analysis section, the system design, and the object design.

The analysis section of OMT builds three separate models using three different notations. The development of these models is an iterative process throughout OMT.

The first analysis model is the Object Model which includes diagrams built with a notation that expands entity relationship modeling to include entity operations. This phase describes the structure of the system's objects and the relationships among them. The object model represents the static structural aspects of the system.

The second analysis model is the Dynamic Model. Its purpose is to capture the essential dynamics of the system. Every object identified in the Object Modeling process has a Dynamic Model constructed using State Transition Diagrams. These are used to describe

the control processes of the system. This model represents the behavioral and control aspects of the system over time.

The third model in the analysis phase is the Functional Model. The Functional Model has the highest level of abstraction and is used to describe the computations within the system and the functionality of the system.

The analysis phase starts with the development of a written description of a problem statement for the problem domain before the three modeling techniques are applied. The Object Model is built by identifying object classes, class associations, object and link attributes, and inheritance. The nouns in the problem statement usually identify the object classes in the system while the verbs or verb phrases in the problem statement can be used to identify associations among classes. Classes during the Object Model are also organized and simplified using inheritance.

The other two analysis phase models are then developed. The Dynamic Model is developed by identifying interaction sequences, event flow diagrams for the system, and a state diagram for each class exhibiting important dynamic behavior. The Functional Model is then constructed by identifying constraints and input and output values, creating data flow diagrams for functional dependencies and writing functional and optimization descriptions.

The second phase of OMT is the System Design phase. This phase encompasses the complete design of the system's architecture with an organized collection of subsystems, a description of the data structures, a decision on control structures, and other system design fundamentals.

The system design phase begins by organizing the system into a series of subsystems which are also allocated to system resources. The basic data structure is developed including the mechanisms for controlling data. Other steps during this phase include identifying concurrency considerations and selecting an approach to software control implementation.

The third OMT phase is the Object Design phase. This phase takes the object structure and refines these objects by incorporating information from the Dynamic and Functional Model stages into object operations.

The Object Design phase starts by combining the three models from the analysis phase to obtain object operations. These operations consist of each event in the Dynamic Model and each process in the Functional Model. Next, algorithms and data structures are developed for these operations followed by the optimization and implementation of class structures, associations, and software control.

Two key object-oriented concepts as defined by this method are

> *Object*: Concept, abstraction, or thing in the world with crisp boundaries meaningful for the problem at hand to promote understanding of the real world and provide an asis for computer implementation.
>
> *(Object) Class*: Group of objects with similar properties (attributes), behavior (operations), common relationships to other objects, and common semantics.

2.3.7 Booch (1991)

Booch's evolution in object technology continued with the publication of his 1991 book, considered by many to be essential reading for those examining object-oriented technology. A second edition was published in 1994.

The most difficult part of object-oriented analysis and design, according to Booch, is the identification of objects and classes. The classification process is the most fundamental issue in object-oriented analysis and design. He is also very concerned with the complexity of software systems and the methods to control and manage complexity. He stresses the importance of the structure of complex systems.

Booch proposes four models for object-oriented development: the logical and physical structures of a system and its static and dynamic semantics. Booch differentiates between the logical and physical structures of a system; and within each structure, a static and a dynamic model are developed. His method is an iterative one with the continual refinement of the logical and physical frameworks of the system.

Two processes are described for the object-oriented development process: a macro and a micro process. The macro process, which controls the micro process, develops core system requirements, models desired system behavior, creates an architectural design, and provides for an evolutionary implementation. The micro process is the more concrete activity driven by and necessitated by the macro process work product. The tasks in the micro process include identifying classes, objects, and the semantics of these classes and objects. The micro process also identifies the relationships among the classes and objects as well as specifying the interface and implementation of the classes and objects.

To capture the logical static description of the system, Booch uses object diagrams for displaying objects and object relationships. Class Diagrams are used similarly to show the class structure and the relationships among classes. The logical dynamic view of the system uses State Transition Diagrams to describe an object's states and the transitional events and resultant activities. Interaction Diagrams are developed to detail any dynamics or interactions among the objects in the object diagrams. The physical static view of the system uses module and physical diagrams for designing and allocating the class and object modules to processes and processors.

2.3.8 Object-Oriented Software Engineering (OOSE) (Jacobson, 1992)

Jacobson developed the use case approach to object-oriented design and analysis, and it forms the integral part of OOSE. Larry Constantine captured the essence of OOSE in his foreword to Jacobson's book. "His approach centers on an analysis of the ways in which a system is actually used, on the sequences of interactions that comprise the operational reality of the software being engineered."

The basis for OOSE originates from three totally different techniques: object-oriented programming, conceptual modeling, and block design. The object-oriented programming technique supplies OOSE with the object-oriented concepts of encapsulation, inheritance, and the relationships among classes and instances. Conceptual modeling is used to develop an understanding of the system and to construct the system architecture. Block design, originating in the telecommunications industry, is used to provide diagrams of the software modules and functionality along with the connections and interfaces among them.

OOSE works with five different models to model system development, each of which attempts to capture some part or aspect of the system by focusing only on that part of the system. These models are

1. Requirements model.
2. Analysis model.
3. Design model.
4. Implementation model.
5. Test model.

The requirements model attempts to capture the functional requirement of the system from a user perspective and focuses on how a potential user would use the system. This is accomplished by the development of three sub-models 1) a use case model, 2) an interface description, and 3) a problem domain model.

The use case model is the most vital part of Jacobson's methodology and is the main concept associated with OOSE and, as such, has emerged in other methods (Rumbaugh, 1994; Graham, 1994; Wilkinson, 1995). The main idea of the use case model is to describe how users interact with and use the system. It utilizes the concept of actors to define what exists outside the system that impacts the system and the idea of use cases to describe what exists and should be done by the system. Actors represent what interacts with the system and define roles that users play in exchanging information, in the form of a dialogue, with the system. Users of the system are instances of actors as actors are defined as a class and the user as an instance of the actor class. The user is the actual user of the system while the actor represents a certain role that a user can play (Jacobson, 1992).

Jacobson describes two kinds of actors, primary and secondary. The primary actors are those who are going to use the system directly while secondary actors are those actors who are supervising and maintaining the system. Secondary actors exist and operate to keep the system usable for the primary actors. As the system structure must reflect the user's perspective, the identification of the use cases start with the primary actors. This will keep the development of the system as close as possible to the most important users. Actors are identified first as they are the major tool for determining the use cases.

The instance of the actor, the user, performs a number of operations on the system. This sequence of operations or transactions are related in a behavioral sense and form the use case. The use case incorporates these related series of user-system interactions into a conceptual whole. Each use case is a specific way of using the system, and every execution of the use case may be viewed as an instance of the use case (Jacobson, 1992). The user may trigger a use case by interacting with the system, causing the use case to initiate one or more activities associated with the particular use case. The use case is a complete description of the events specifying all the actions between the user and the system, and a particular use case instance will exist only as long as the use case is operating or performing the transactions or activities of the use case.

The use cases are identified through the actors. The actor's perspective of the system is determined and may include questions such as

1. What are the main tasks of each actor?
2. Will the actor have to read or write or change any of the system information?

3. Will the actor have to inform the system about outside changes?

4. Does the actor wish to be informed about unexpected changes?

The use case model uses a diagram where the system is represented by a boundary box. Actors are identified as persons outside the box while use cases are represented as ellipses inside the box. An illustration of a generic use case is found in Figure 2. The use case model is described by the actors and the use cases in the model, together with the associated relationships among these entities in the system.

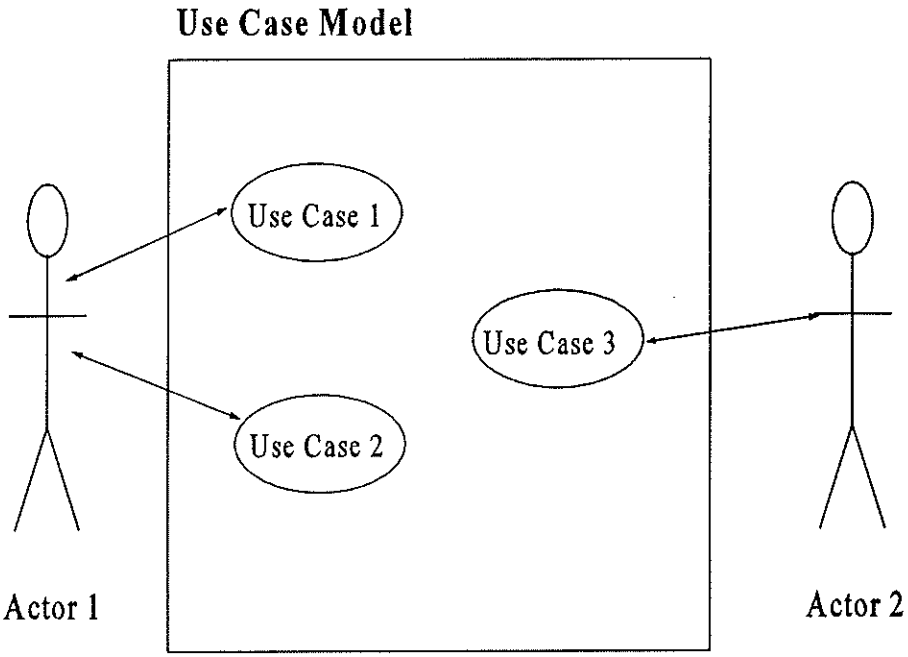


Figure 2. Use Case Model Diagram Example.

The first part of a use case to be described is the basic course. The basic course details the most important course of events that give the best understanding of the use case. Each use case has one basic course, but may also have several alternative courses.

Alternative courses are possible variants of the basic course and include any possible errors and error handling that may occur during the use case.

Extension may be used to structure and relate use case descriptions. Extension stipulates how one use case may be inserted into another use case, thereby extending the use case description. This allows for greater flexibility and easier modification of the system by providing a mechanism for structuring relationships and associations among the use cases. During the operation of a use case, any extension by another use case occurs at the point of insertion. After the extended use case has completed its activity, the original use case continues and completes its operation.

Jacobson calls this part of the analysis and design use case driven design. The design of the whole system is driven and controlled by how the users use and make use of the system. Changes and modifications to the system are accomplished by remodeling the actors and/or the use cases.

The other two sub-models of the requirements model are created to support the use case model. The interface description captures the user's logical view of the system in an attempt to make the system behavior consistent with the user's logical system view. Jacobson suggests a prototype of the user interface as the perfect tool for eliciting the user's logical view of the system and refining the user interface. The problem domain model is developed to analyze objects in the problem domain which will have a counterpart in the system.

The use case model is used to develop the other models in the analysis and drives the development of these models (Figure 3). The other models detail various sections of system

development. The analysis model provides structure to the system through the use of three types of objects: 1) interface objects, 2) entity objects, and 3) control objects.

The Design Model refines the Analysis Model and aims to adopt and refine the object structure to the current implementation environment. The Implementation Model proposes to implement the system and, lastly, the Test Model attempts to verify the system design by testing the Implementation Model.

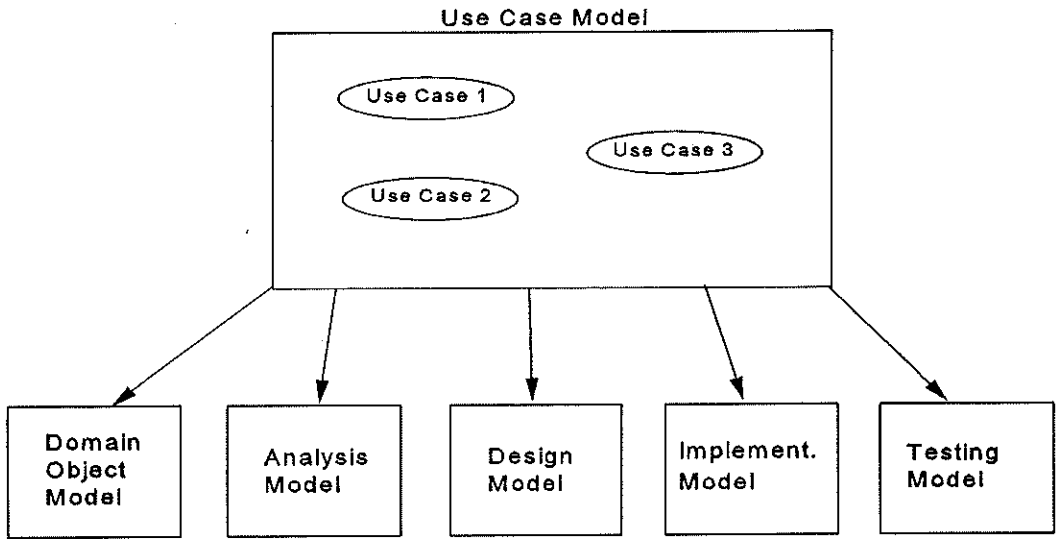


Figure 3. Use Case Model Drives the Other Analysis and Design Models.

2.3.9 Booch (1994)

The second edition of Booch's seminal work was published in 1994. In this edition, Booch states that the two primary activities of analysis are domain analysis and scenario planning. In domain analysis, the objects and classes of a particular problem domain are identified. The scenario planning, an addition to the 1994 edition, is described as the central activity in analysis.

Scenario planning is used to identify and develop all the fundamental behaviors of the system. Booch describes the first two steps of scenario planning as:

1. Identify all the primary function points of the system, and, if possible, group them into clusters of functionally related behaviors. Consider also clustering according to hierarchies of functions, wherein certain high-level functions build upon more primitive ones.
2. For each interesting set of function points, storyboard a scenario, using the techniques of use-case and behavior analysis briefly described by Booch in his Chapter 4. CRC card techniques are effective in brainstorming about each scenario. As the semantics of each scenario become clearer, document them using object diagrams that illustrate the objects that are initiators or contributors of behavior and that collaborate to carry out the activities of the scenario. Include a script that shows the events that trigger the scenario and the resulting order of actions. In addition, document any assumption, constraints, or performance issues for each scenario.

The remaining steps build on and refine these first two steps.

Scenario planning includes use case or behavior analysis techniques as part of the second step. This is an addition to Booch's 1991 method and recognizes the use case as a valuable part of the central analysis activity. While the use case is not the main driving point of the analysis, it is included in the major analysis activity.

2.3.10 Other Use Case Methods and Applications

Other examples of the use and extension of Jacobson's use case model include combining the main use cases created during a requirements analysis with early rapid prototyping (Hansen and Miller, 1995). This approach provides the user with a graphical representation of the main use case scenarios early in the development process. The user feedback generated during this process provides the designers with an opportunity to assess and verify the main use case behavior as well as providing information about designing the other use cases in the system.

American Management Systems (AMS) uses an approach that extends the use case model through the use of several levels of use cases, functional area models, and use case dependency diagrams (Armour, Boyd, and Sood, 1995). AMS found that a single, flat layer or level of use cases was not sufficient when modeling large business systems. AMS expands and stratifies the use case model to include four primary use case levels:

1. High level use cases.
2. Expanded use cases.
3. Detailed use cases.
4. Abstract use cases.

High level use cases identify and describe broad system behaviors initiated by the actors associated with the system (Armour, Boyd, and Sood, 1995). The expanded use case is a refinement of the high level use case and is similar to the basic course of the Jacobson method. The expanded use case level focuses on the fundamental activities of the high level use case behaviors. In the third level, the detailed use case, detailed descriptions of system

functionality are developed, including exception conditions and alternative courses like Jacobson's. The last level, the abstract use case, describes common functionality that is utilized by multiple use cases.

AMS uses a functional area model to organize high level use cases into functional areas to provide a framework for understanding the system's functional architecture. In their business modeling, AMD organizes use cases into functional areas based upon business activities whereby system behavior is displayed from a functional viewpoint. This provides the analyst a look at the major functions supported by the system together with the activities necessary to support those functions.

AMS also develops a use case dependency diagram. The use case dependency diagram, as outlined by AMS, provides:

1. Enhanced understanding of system functionality and scope, by organizing system functionality into a life cycle of events and states.
2. Increased traceability by helping to map use cases to any work flow models or business scenarios previously developed.
3. Identification of missing use cases. A use case with a precondition that is not met by the execution of another use case may indicate a missing use case.

Among others considering use cases is Berard (1995). Berard, while pointing out the value of use cases, cautions software engineers to beware of the difference of the functional aspects of use cases and the object-oriented approach. Use cases describe the system in terms of how the user will see it and use it and in terms of what is delivered to the

uses. This is basically a functional view of the system, and functional approaches localize information around functions while object-oriented approaches localize information around objects. Berard advises that functions and object do not map directly to one another so the underlying object-oriented architecture of a system should not directly reflect the use case, functional view of the system. He says that the creation, integration, and maintenance of use cases should be carefully controlled to avoid problems in object duplication among several use cases, localization of information into object definitions not functional definitions, and maintaining information hiding in system objects.

2.3.11 Unified Method (Expected 1996)

The Unified Method is expected to be published sometime during the summer of 1996 (Booch and Rumbaugh, 1995). It is the unification of Rumbaugh's OMT method with the Booch method and the incorporation of important ideas from other methods. It should be noted and stressed that Jacobson's use case will be a part of the model, with only minor changes to better match the overall approach (Booch and Rumbaugh, 1995).

The Unified Method, along with Jacobson's use case model, represents a large majority of the users of object technology. As such, it has the potential to become the standard for object-oriented analysis and design. Draft version 0.8 is available from Rational Software Corporation and describes the method as a third generation object-oriented analysis and design method developed by Grady Booch and Jim Rumbaugh from the unification of the Booch and OMT methods (Booch and Rumbaugh, 1995). The method is scheduled to be designed so that Booch and OMT users can both move into the new, unified method. Because of its preliminary nature, the discussion of the Booch and

OMT methods and this paper's interest in user centered object-oriented analysis and design, the only part of the method presented here is the use case model. The discussion in the draft of the Unified Method indicates that the method will rely upon Jacobson's use cases.

The Unified Method defines a use case as a generic description of an entire transaction involving several objects and a use case model as the collection of use cases specifying or characterizing the behavior of a whole application system together with one or more external actors that interact with that system (Booch and Rumbaugh, 1995). The use case's meaning can usually be an informal text description of the external actors and the sequence of events among the objects in the system providing the use case's functionality. The use case, as such, will be a description of particular behavior in the system at a high level of abstraction. The use case model protrays the entire system as the collection of use cases at the top level of abstraction.

A use case diagram like Jacobson's is used to display the complete system and its use cases associated with their actors. Use cases that utilize other use cases are denoted, and a box surrounding the use cases acts as the boundary between the system and the actors.

The Unified Method authors conclude their discussion with the note that inheritance and aggregation relationship as a theoretical concept will be explored further with Jacobson.

2.4 Summary

The object-oriented analysis and design methods presented in this chapter feature some of the most important contributions in the development of object-oriented technology. Taken together, they trace the evolution of object-oriented technology (Table 3).

Table 3. A Brief Summary of the Object-oriented Methodologies Under Consideration.

Method and Year Introduced	Major Feature or Contribution
Abbott, 1983	Use of natural language problem descriptions. Adopted by many other methods, not object-oriented
Booch, 1986	Oldest of Object-oriented methods. Based on Ada.
GOOD, 1986	Based on Ada.
HOOD, 1989	Two kinds of objects, passive and active. Two types of hierarchies, compositional and usage.
Coad and Yourdon, 1991	First widely published, complete object-oriented method. Enumerated abstraction, encapsulation and inheritance.
OMT, 1991	Regarded as a very complete method and notation. Widely used by developers.
Booch, 1991	Classic object-oriented description. Popular method.
Jacobson, 1992	Use case model introduced. Use cases adopted by other methods.
Booch, 1994	Scenario planning part of central analysis activity. Use case suggested as one way for scenario planning.
AMS, Berard; 1995	Expand use cases, caution of use case use.
Unified Method, 1996	Merging of two dominate methods, OMT and Booch. Add use case model to method.

2.5 Conclusion

This chapter surveyed and examined several major object-oriented analysis and design methodologies to accomplish a major objective of this paper, selecting a user centered object-oriented analysis and design methodology. After this evaluation, it is the conclusion of this paper that selecting Jacobson's use case driven approach achieves that objective.

Jacobson's use case driven approach immediately centers on the user and remains centered on the user throughout the analysis and design process. This approach develops its first concepts and models of the system from the user's perspective and uses them to drive all the other analysis and design in the process. It develops use cases, at the start of the analysis, that define the system from the user's perspective, the user's perception on how the user will utilize the system and what the user will do with the system. This method's emphasis is on the user, and that emphasis drives the entire analysis. Jacobson's use case driven approach remains user centered throughout system analysis and design.

Use cases center all analysis and design on the user and have no counterpart driving system analysis in the other methodologies. Coad and Yourdon's human interaction component occurs after the completion of object-oriented analysis and does not address the issue of the user's perspective of using the system as part of system analysis. They do, importantly, use their human interaction component to identify user characteristics and work task scenarios, but, again, this is after object-oriented analysis. They recommend that developers incorporate human-computer guidelines in the development of a system

prototype, but this is during the design phase and is used only for interface development, not system analysis.

The Booch method and Rumbaugh's OMT have been the leading methodologies in the field and are merging into the Unified Method. The Booch 1994 method suggests using use cases as part of its central analysis activity, scenario planning. The Unified Method will include Jacobson's use case model which is instructive about the evolution of their current approaches.

Rumbaugh (1994) had this to say about use cases in 1994: "Most methodologists now agree that user-centered analysis is the best way to solve the right problem. Capturing the user's needs is a major focus of several methodologies, including Rubin and Goldberg's OBA and Jacobson's Objectory. In particular, Jacobson's use cases have been well received by just about every methodologist including us."

Nancy Wilkinson, writing in November of 1995 (Wilkinson, 1995), says this about Jacobson's use case methodology: "Most important to the methodology is, of course, the notion of stressing uses of the system to build reusable and adaptable applications. Use cases have been so well received that other methodologists have begun to incorporate them in their methods and notations."

CHAPTER 3. IMS USER CENTERED SYSTEM ANALYSIS

3.1 Introduction

This chapter applies Jacobson's use case driven approach to system analysis to the IMS. The analysis draws on interviews with the NDDOT IMS coordinator, discussions with a transportation research professional, an interim IMS report, and background surveys (Schmidt, 1996; Tolliver, 1996). These were used to capture and provide a user centered framework for systems analysis. The chapter uses Jacobson's use case methodology described in Chapter 2 and augments the use case concept with the AMS stratification, also described in Chapter 2. The study also introduces an extension to Jacobson's actor identification.

The chapter starts out with Jacobson's requirements model, the first model in his methodology, which initiates actor identification and use case development. Jacobson's actor identification determines the primary and secondary actors in the system, and this paper extends that concept with a priority ranking of the primary and secondary actors identified in the system. The use case development is delineated into high level use cases as developed by AMS and is followed with an example of a high level use case detailed into an expanded use case. This is followed by a section on object specification of the example use case which includes descriptions and diagrams of the objects as well as a narrative of the example use case with objects included. This section also discusses the inheritance and encapsulation of the example objects, and the reuse of the example objects by the other high level use case identified in the system. A comparison is made between the user centered,

object-oriented analysis adopted in this paper with a data-driven approach. The chapter closes with a discussion about the interface description of the IMS.

3.2 Requirements Model

The requirements model is developed to delineate the system and define the functionality supported by the system. It is the first model developed in the methodology; and its central feature and the central feature of the entire analysis is the use case. The requirements model drives the subsequent models which include the analysis model, design model, implementation model and testing model (Jacobson, 1992). It is central to the development of the system, and all other models are verified against the requirements model. The requirements model is developed from the users' perspective and includes 1) actor identification, 2) use case development, 3) interface descriptions, and 4) problem domain objects. The fourth is not discussed because of the more standard notation adopted for the object diagrams (OMT).

3.2.1 Actor Identification

The identification of the IMS actors involved interviews with the NDDOT IMS coordinator, an interim IMS report, and background surveys with a NDDOT transportation planner (Schmidt, 1996). The first survey narrowed down the potential users in the NDDOT while the second survey delineated among primary and secondary actors as described by Jacobson. The interviews identified a significant gap between the expected level of use by the transportation planners and the other actors. Transportation planners are going to be the most important users of the system. The second survey was designed to address this issue and extended Jacobson's actor identification by adding a priority ranking of the actors.

This priority ranking is particularly useful when one actor dominates the expected use of the system. The rankings in the survey are based upon an assessment by the NDDOT IMS coordinator.

The primary actors in the IMS, ranked in order of the importance of their use of the system, are 1) Transportation Planners, 2) Mapping Department, 3) Rail Program, and 4) Geographical Information System (GIS). The first three of the primary actors are humans while the fourth primary actor is a computerized system. The human actors have exclusive priority over the computerized system actors. The secondary actors, ranked on the same scale as the primary actors, are 5) Traffic Operations, 6) Upper Management, 7) Planning Division Head, 8) Operations, 9) Program and Project Development, 10) Secondary Roads, 11) Pavement Management System, 12) Transportation Data Systems, and 13) Traffic Operations Systems.

The transportation planners were identified as the main users of the IMS. In fact, their use is expected to dominate the system and, as such, will receive the most consideration during the analysis and design process. The other 12 users, of which three are primary users, will receive less attention. In that context, the following discussion will concentrate exclusively on the transportation planner.

3.2.2 High Level Use Case Examples

This section presents several high level use cases of the IMS. The use case stratification of the AMS was used in a survey to stratify background use case information into 1) high level use cases, 2) expanded use cases, 3) detailed use cases, and 4) abstract use cases. The use cases presented here are high level use cases.

A series of interviews, meetings, an interim IMS report, and surveys were used to develop these preliminary high level use cases. The interviews, meetings, and report identified several major, high level use cases, and the surveys provided context for the functionality involved in these use cases.

The IMS high level use cases presented here are 1) Intermodal Facility Efficiency, 2) Estimate Highway Investment Costs, 3) Railroad Branchline Rehabilitation Evaluation, 4) Grain Elevator Modal Split, and 5) Fertilizer Distribution Facility Location. A sixth high level use case, Railroad Branchline Abandonment, is presented in greater detail in Section 3.2.3. Figure 4 displays the six high level use cases.

1. *Intermodal Facility Efficiency* is started when *transportation planner* wants to evaluate a particular intermodal facility. When *transportation planner* has selected the facility from a graphical display of intermodal facilities, the system will present the performance measures related to that facility in a form selected by *transportation planner*.
2. *Estimate Highway Investment Costs* is used when the *transportation planner* wants to generate a cost estimation of investing in the highway infrastructure associated with an existing or new intermodal facility. *Transportation planner* will assign the facility location in reference to the highway system, and the IMS will calculate a cost estimation for investing in each section of highway affected by the facility.

3. *Railroad Branchline Rehabilitation Evaluation* is used when the IMS *transportation planner* wants to generate a benefit-cost analysis of investing in the rehabilitation of a railroad branchline. *Transportation planner* will select the railroad branchline for analysis and the IMS will calculate the benefits and costs to the transportation network associated with improving the selected railroad branchline.

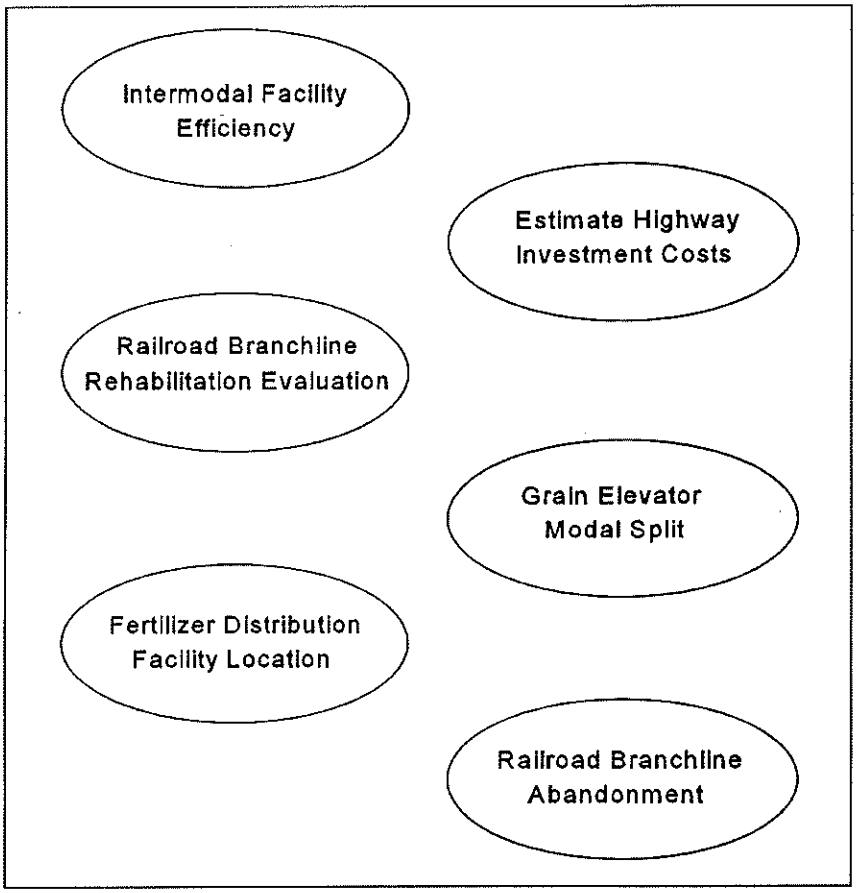


Figure 4. High Level Uses Cases of the Intermodal Management System.

4. *Grain Elevator Modal Split* is initiated when *transportation planner* wants to examine any changes in the modal split between rail and truck grain traffic. *Transportation planner* will interact with the IMS to select the grain elevator or elevators, and the system will return the grain traffic characteristics by mode of transportation.

5. *Fertilizer Distribution Facility Location* is started when *transportation planner* needs to inspect and survey the location of fertilizer distribution centers. The system will provide a graphical display outlining all fertilizer distribution centers in the state in relation to the state highway system.

3.2.3 Railroad Branchline Abandonment Use Case

A major use of the IMS will be the railroad branchline abandonment use case scenario. In this use case, the transportation planner initiates an analysis to calculate the impacts that an abandonment of a railroad branchline will have on the state highway system. Railroad branchline abandonment diverts traffic from the railroad to the highway system and has an impact on the maintenance and traffic characteristics of the highway system. This use case was identified and elaborated by using surveys, interviews, and an interim IMS report mentioned. It is presented here to illustrate the application of user centered object-oriented analysis to the IMS as well as the complexity of the IMS and the design issues involved in developing the system.

The use case is summarized as follows:

Railroad Branchline Abandonment is started by the *transportation planner* when he wants to evaluate what impacts will occur to the highway system when a railroad branchline is abandoned. The *transportation planner* will inform the IMS that he wants a railroad abandonment analysis, and the system will respond with a display of the railroad network. After the *transportation planner* has selected the branchline to be abandoned from the railroad network, the system will calculate the highway impacts specific to the abandonment and return a series of reports to the *transportation planner*.

3.3 Railroad Branchline Abandonment Expanded Use Case

Jacobson's basic course of a use case provides greater detail and traces the most important system events giving a better understanding of the use case. AMS's comparable decomposition is the expanded use case, the expansion of the high level use case into greater detail. The expanded use case for the Railroad Branchline Abandonment use case is as follows:

When the *transportation planner* wants to start the *Railroad Branchline Abandonment* use case, he selects an icon supplied by the IMS for this use case. When selected, the icon launches a full screen window displaying the railroad network, the highway network, and the grain elevator and fertilizer distribution intermodal facilities. The *transportation planner* chooses a railroad branchline to analyze by clicking on the endpoints of a branchline segment on the railroad network. The screen display changes to highlight the selected branchline and the potentially affected portions of the highway network. At this point, *transportation planner* may do two things: 1) terminate the preliminary analysis

process or 2) start the analysis process. If *transportation planner* terminates the process, he is returned to the first screen display in *Railroad Branchline Abandonment* (the railroad network, the highway network, and the grain elevator and fertilizer distribution intermodal facilities). If *transportation planner* starts the analysis process, the following sequence of system events occur:

1. The system determines if a valid branchline scenario has been selected. If a valid branchline scenario has not been selected, the system terminates the analysis, informs the *transportation planner* via a dialogue box, and returns to the first screen display. If a valid branchline scenario has been selected, the *transportation planner* is informed via an informational box that an analysis scenario has started. The informational box tracks the progress of the analysis informing *transportation planner* as the process continues.
2. The analysis process creates a data structure accessing the Pavement Management System and a default highway impact database. The *transportation planner* is presented a dialogue box allowing 1) edit assembled data, 2) proceed with analysis, and 3) terminate analysis. If *transportation planner* chooses to edit the data, a window appears with the highway impact data for this analysis whereby *transportation planner* may edit the data or return to the last dialogue box. If *transportation planner* terminates the process, the system terminates the analysis, informs the *transportation planner* via a dialogue box,

and returns to the first screen display. If *transportation planner* proceeds with the analysis, a check is done on the data structure to ensure that the data going to the calculation objects are within valid ranges. If any invalid data are detected, the IMS suspends execution, notifies *transportation planner* of the invalid data via a dialogue box, and waits for a decision from *transportation planner*. The dialogue box presents *transportation planner* with the invalid data, the reason why it is considered by the system to be invalid, a possible result of continuing with the invalid data, the location of the invalid data, and the choice of either continuing the analysis process with the invalid data or terminating the analysis. If *transportation planner* terminates the process, the system terminates the analysis, informs the *transportation planner* via a dialogue box, and returns to the first screen display. If *transportation planner* decides to continue with the analysis process or no invalid data has been detected, the analysis process proceeds to step 3.

3. The calculation of the highway impacts is performed. This consists of the following steps and is done by the IMS.

   3.1 Compute ESAL lives of impacted highways by functional class.

   3.2 Compute portion of highway deterioration attributable to traffic.

3.3 Compute average cost per ESAL of impacted highways by functional class.

3.4 Compute average ESALs per VMT by truck type, highway class, and pavement type.

3.5 Compute weighted average truck length of haul and backhaul factor.

3.6 Compute incremental ESALs by truck type, highway class, and pavement type.

3.7 Sum incremental ESALs across pavement types.

3.8 Compute annual incremental pavement costs by highway class and pavement type.

3.9 Compute highway costs and ESAL factors.

3.10 Generate highway impact reports.

4. When the analysis process has been completed, *transportation planner* is presented with a full screen display of the geographical area involved in the analysis with the highway network highlighted according to the results of the analysis. *Transportation planner* also has available a report in WordPerfect format containing graphs, data categories, and values returned by the analysis and a template narrative explaining the various components of the report. The report graphs are also available for screen display in sizeable windows through a pop-up menu structure. At this point, *Railroad*

Branchline Abandonment has been completed, and *transportation planner* may close *Railroad Branchline Abandonment* or remain at the first screen display in *Railroad Branchline Abandonment.*

3.3.1 Object Specification of the Railroad Branchline Abandonment Use Case

This section presents the specification of the objects derived from the Railroad Branchline Abandonment (RBA) Expanded Use Case narrative. The discussion includes a description of the objects and their roles in the use case. Figure 5 shows a diagram of the object specifications starting with the RBA Icon object while Figure 6 displays the object specifications starting with the HIA Start Window object.

1. RBA Icon: The icon represents the interface to the use case and will initiate the RBA use case. It has methods for showing itself and displaying the starting window in the analysis process. It knows its screen position, icon image, and display parameters. Its role is to provide access to the use case.
2. HIA Window: This object presents a display of the railroad network, the highway network, the grain elevators, and the fertilizer distribution centers. It has several methods for showing itself, showing or hiding the transportation networks and intermodal facilities, and showing the results of the analysis. It also has methods for starting or terminating analysis. Its role is to display the state of the analysis process and to act as a central point for message passing among the objects.

Figure 5. Railroad Branchline Abandonment Use Case Object Specification, Part I

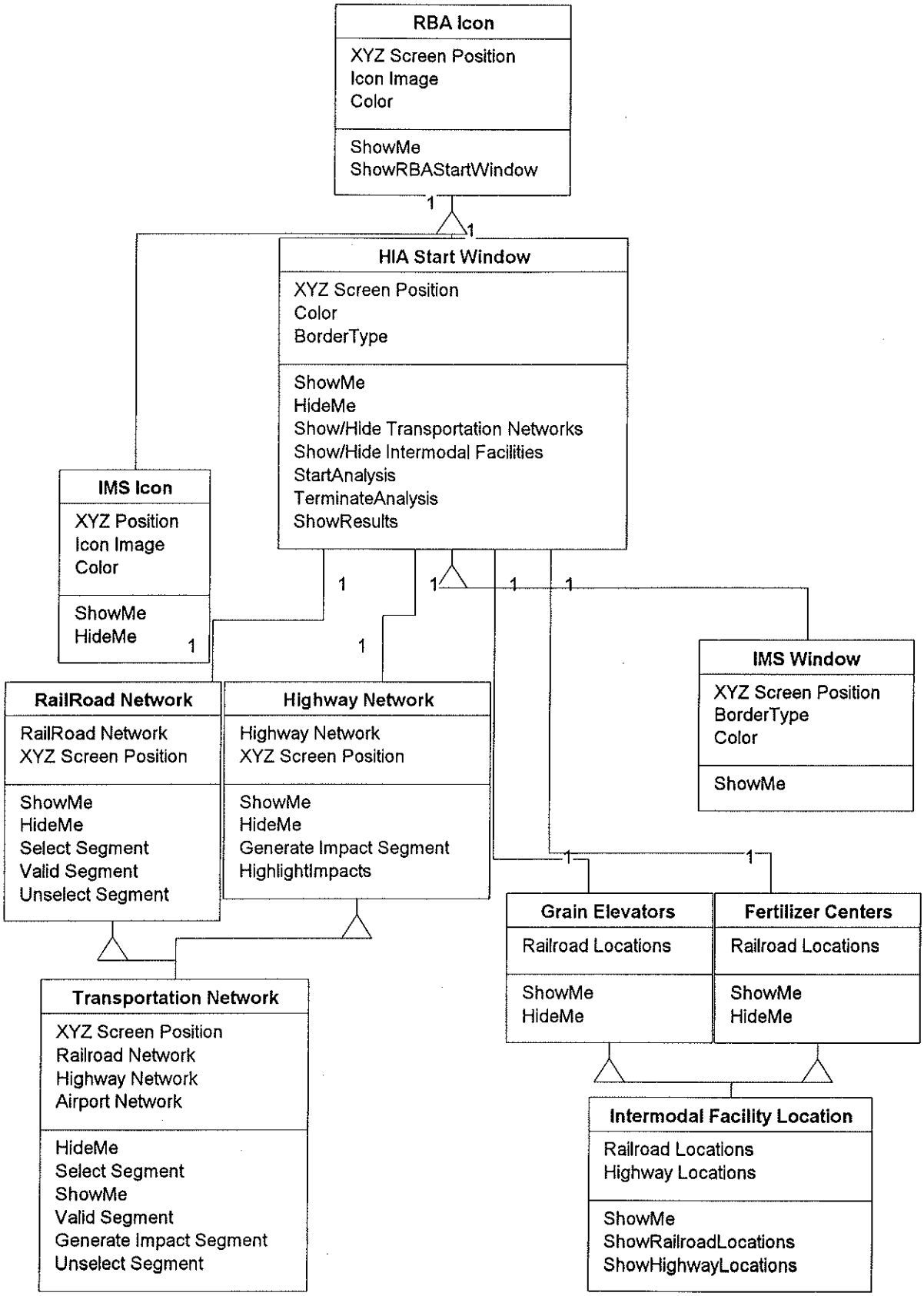
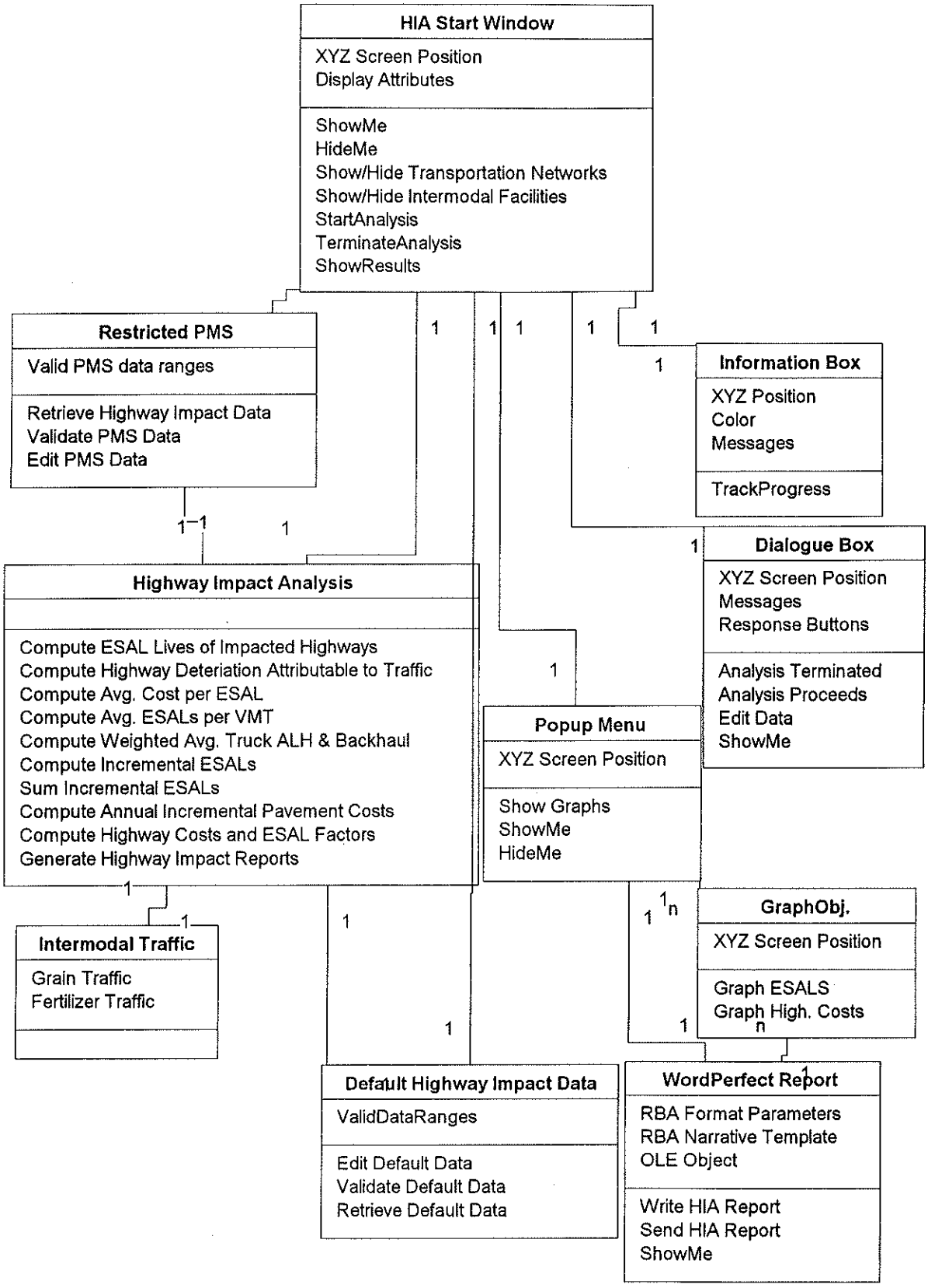

RBA Icon
XYZ Screen Position
Icon Image
Color
ShowMe
ShowRBAStartWindow
HIA Start Window
XYZ Screen Position
Color
BorderType
ShowMe
HideMe
Show/Hide Transportation Networks
Show/Hide Intermodal Facilities
StartAnalysis
TerminateAnalysis
ShowResults
IMS Icon
XYZ Position
Icon Image
Color
ShowMe
HideMe
IMS Window
XYZ Screen Position
BorderType
Color
ShowMe
RailRoad Network
RailRoad Network
XYZ Screen Position
ShowMe
HideMe
Select Segment
Valid Segment
Unselect Segment
Highway Network
Highway Network
XYZ Screen Position
ShowMe
HideMe
Generate Impact Segment
HighlightImpacts
Grain Elevators
Railroad Locations
ShowMe
HideMe
Fertilizer Centers
Railroad Locations
ShowMe
HideMe
Transportation Network
XYZ Screen Position
Railroad Network
Highway Network
Airport Network
HideMe
Select Segment
ShowMe
Valid Segment
Generate Impact Segment
Unselect Segment
Intermodal Facility Location
Railroad Locations
Highway Locations
ShowMe
ShowRailroadLocations
ShowHighwayLocations
1

Figure 6. Railroad Branchline Abandonment Use Case Object Specification, Part II


HIA Start Window
XYZ Screen Position
Display Attributes
ShowMe
HideMe
Show/Hide Transportation Networks
Show/Hide Intermodal Facilities
StartAnalysis
TerminateAnalysis
ShowResults
Restricted PMS
Valid PMS data ranges
Retrieve Highway Impact Data
Validate PMS Data
Edit PMS Data
Information Box
XYZ Position
Color
Messages
TrackProgress
Dialogue Box
XYZ Screen Position
Messages
Response Buttons
Analysis Terminated
Analysis Proceeds
Edit Data
ShowMe
Highway Impact Analysis
Compute ESAL Lives of Impacted Highways
Compute Highway Deteriation Attributable to Traffic
Compute Avg. Cost per ESAL
Compute Avg. ESALs per VMT
Compute Weighted Avg. Truck ALH & Backhaul
Compute Incremental ESALs
Sum Incremental ESALs
Compute Annual Incremental Pavement Costs
Compute Highway Costs and ESAL Factors
Generate Highway Impact Reports
Popup Menu
XYZ Screen Position
Show Graphs
ShowMe
HideMe
Intermodal Traffic
Grain Traffic
Fertilizer Traffic
GraphObj.
XYZ Screen Position
Graph ESALS
Graph High. Costs
Default Highway Impact Data
ValidDataRanges
Edit Default Data
Validate Default Data
Retrieve Default Data
WordPerfect Report
RBA Format Parameters
RBA Narrative Template
OLE Object
Write HIA Report
Send HIA Report
ShowMe

3. Railroad Network: The Railroad Network object knows the railroad network and displays this inside the HIA Window. It has methods for selecting/unselecting a railroad branchline segment and for validating that segment as well as for showing and hiding itself. This object is inherited from the transportation network object. Its role is to provide a mechanism for the user to select a railroad and to provide visual cues of the state of the railroad network.

4. Highway Network: The highway network object knows the highway network and displays this inside the HIA Window. It has methods for selecting a highway segment, for generating a highway impact segment, and for validating the selected highway segment.This object is inherited from the transportation network object. Its role is to display and provide visual cues of the state of the railroad network.

5. Grain Elevators: This object is inherited from the Intermodal Facility object, knows the railroad locations of the grain elevators, and has the method ShowMe. Its role is to provide a visual cue to the user of the location of grain elevators on the railroad network.

6. Fertilizer Centers: This object is inherited from the Intermodal Facility object, knows the railroad locations of the fertilizer centers, and has the method ShowMe. Its role is to provide a visual cue to the user of the fertilizer center location on the railroad network.

7. Dialogue Box: This object is inherited from the IMS dialogue box object (not displayed in the accompanying diagram) and has methods ShowMe, Analysis Terminated, Analysis Proceeds, and Edit Data. Its role is to inform the user of events or changes in the system requiring user interaction.

8. Information Box: This object is inherited from the IMS dialogue box object (not displayed in the accompanying diagram) and has the TrackProgress method. Its role is to inform the user of events or changes in the analysis process or system.

9. Restricted PMS Object: The role of this object is to retrieve, edit, and validate data from the Pavement Management System. It knows the proper data types and ranges for the PMS data. The data will be used by the Highway Impact Analysis.

10. Default Highway Impact Data Object: This object retrieves and validates default data used in every Highway Impact Analysis. The data may be changed by the user. Its role is to provide validated default data for a Highway Impact Analysis.

11. Highway Impact Analysis Object: This object has the calculation methods for performing a Highway Impact Analysis. The data may be changed by the user. Its role is to perform the calculations necessary for the Highway Impact Analysis.

12. RBA WordPerfect Report Object: This object provides a WordPerfect document containing graphs, data categories, and values returned by the analysis, and a template narrative explaining the various components of the report. Its role is to supply the necessary reporting formats requested by the user.

3.3.2 Railroad Branchline Abandonment Use Case With Objects

The following narrative presents the description of the Railroad Branchline Abandonment Use Case provided in Section 3.2.4 and inserts the *Railroad Branchline Abandonment* objects and methods into the narrative. This is done to connect the narrative with the objects required in the system. It provides an enhanced textual description of the use case and addresses a concern of some methodologists that textual descriptions need to be closely linked to object specification. The object and methods are inserted where they would exist in an actualized Railroad Branchline Abandonment Use Case. The objects are found in curly brackets {} and are bolded. The object name is displayed and is followed by a method.

The Railroad Branchline Abandonment Use Case is further expanded with objects as follows:

When the *transportation planner* wants to start the *Railroad Branchline Abandonment* use case, he selects an icon supplied by the IMS **{RBA Icon.ShowMe}** for this use case. When selected, the icon **{RBA Icon.ShowRBAStartWindow}** launches a full screen window **{HIA Window.ShowMe}** displaying the railroad network **{Railroad Network.ShowMe}**, the highway network **{Highway Network.ShowMe}**, and the grain

elevator **{Grain Elevators.ShowMe}** and fertilizer distribution intermodal facilities **{Fertilizer Centers.ShowMe}**. The *transportation planner* chooses a railroad branchline **{Railroad Network.SelectSegment}** for analysis by clicking on the endpoints of a branchline segment on the railroad network. The screen display changes to highlight the selected branchline **{Railroad Network.SelectSegment}** and the potential affected portions of the highway network **{Highway Network.GenerateImpactSegment}**. At this point, *transportation planner* may do two things: 1) terminate the preliminary analysis process **{HIA Window.TerminateAnalysis}** or 2) start the analysis process **{HIA Window.StartAnalysis}**. If *transportation planner* terminates the process **{HIA Window.TerminateAnalysis}**, he is returned to the first screen display in *Railroad Branchline Abandonment* (the railroad network, the highway network, and the grain elevator and fertilizer distribution intermodal facilities). If *transportation planner* starts the analysis process **{HIA Window.StartAnalysis}**, the following sequence of system events occur:

1. The system determines **{Railroad Network.ValidSegment}** if a valid branchline scenario has been selected. If a valid branchline scenario has not been selected, the system terminates the analysis **{Dialogue Box.AnalysisTerminated}**, informs the *transportation planner* via a dialogue box, and returns to the first screen display **{HIA Window.ShowMe}**. If a valid branchline scenario has been selected, the *transportation planner* is informed via an informational box **{Information Box.ShowMe}** that an analysis scenario has started.

The informational box tracks **{Information Box.TrackProgress}** the progress of the analysis informing *transportation planner* as the process continues.

2. The analysis process creates a data structure accessing the Pavement Management System **{RestrictedPMS.RetrieveData}** and a default highway impact database **{DefaultHighwayImpactData}**. The *transportation planner* is presented a dialogue box **{DialogueBox.ShowMe}** allowing 1) editing of the assembled data **{DialogueBox.EditData}**, 2) proceeding with the analysis **{DialogueBox.AnalysisProceeds}**, and 3) terminating the analysis **{DialogueBox.AnalysisTerminated}**. If *transportation planner* chooses to edit the data, a window **{DefaultHighwayImpactData.EditData}** appears with the highway impact data for this analysis whereby *transportation planner* may edit the data **{EditDataWindow.DisplayData}** or return to the last dialogue box **{EditDataWindow.HideMe}**. If *transportation planner* terminates the process **{DialogueBox.AnalysisTerminated}**, the system terminates the analysis, informs the *transportation planner* via the dialogue box, and returns to the first screen display **{HIA Window.ShowMe}**. If *transportation planner* proceeds with the analysis **{DialogueBox.AnalysisProceeds}**, a check

{RestrictedPMS.ValidatePMSData}{DefaultHighwayImpactDat a.ValidateData} is done on the data structure to ensure that the data going to the calculation objects are within valid ranges. If any invalid data are detected, the IMS suspends execution **{DialogueBox.ShowMe}**, notifies *transportation planner* of the invalid data via a dialogue box, and waits for a decision from *transportation planner*. The dialogue box presents *transportation planner* with the invalid data, the reason why it is considered by the system to be invalid, a possible result of continuing with the invalid data, the location of the invalid data, and the choice of either continuing the analysis process with the invalid data or terminating the analysis. If *transportation planner* terminates the process **{DialogueBox.AnalysisTerminated}**, the system terminates the analysis, informs the *transportation planner* via a dialogue box, and returns to the first screen display **{HIA Window.ShowMe}**. If *transportation planner* decides to continue with the analysis process **{Dialogue Box.AnalysisProceeds}** or no invalid data has been detected, the analysis process proceeds to step 3.

3. The calculation of the highway impacts is now done. This consists of the following steps and is done by the IMS **{HighwayImpactAnalysis.Methods}**.

3.1 Compute ESAL lives of impacted highways by functional class.

3.2 Compute portion of highway deterioration attributable to traffic.

3.3 Compute average cost per ESAL of impacted highways by functional class.

3.4 Compute average ESALs per VMT by truck type, hwy. class, and pavement type.

3.5 Compute weighted average truck length of haul and backhaul factor.

3.6 Compute incremental ESALs by truck type, highway class, and pavement type.

3.7 Sum incremental ESALs across pavement types.

3.8 Compute annual incremental pavement costs by hwy.class and pavement type.

3.9 Compute highway costs and ESAL factors.

3.10 Generate highway impact reports.

4. When the analysis process has been completed, *transportation planner* is presented with a full screen display **{HIA Window.ShowMe}** of the geographical area involved in the analysis with the highway network highlighted **{HighwayNetwork.HighlightImpacts}** according to the results of

the analysis. *Transportation planner* also has available a report in WordPerfect format **{RBAWordPerfectReport.ShowMe}** containing graphs, data categories and values returned by the analysis, and a template narrative explaining the various components of the report. The report graphs are also available for screen display **{HIAWindow.ShowResults}** through a pop-up menu structure **{PopupMenu.ShowMe}.** At this point, *Railroad Branchline Abandonment* has been completed, and *transportation planner* may close *Railroad Branchline Abandonment* **{HIAWindow.HideMe}** or remain at the first screen display in *Railroad Branchline Abandonment.*

3.3.3 Object Inheritance in the Railroad Branchline Abandonment Use Case

Several of the *Railroad Branchline Abandonment* objects are inherited from other objects. Inheritance is the mechanism whereby objects acquire or inherit the characteristics and behaviors of another object, a parent object. The characteristics and behaviors of the parent object may be added, modified, or even hidden in the child object (Booch 1994). Inheritance is also a form of reuse as one parent object may be used continually to create one or more child objects, thereby creating a hierarchy of objects. This hierarchy allows the inheriting of characteristics and behaviors without programming entirely new entities and also makes possible the process where any changes to the parent object are propagated throughout the system to the child objects.

In *Railroad Branchline Abandonment,* the RBA Icon object is inherited from the IMS object and acquires the IMS icon's display attributes and ShowMe method. The HIA Start Window is inherited from the IMS Window and acquires the IMS Window's display attributes and ShowMe method. The Railroad Network and Highway Network are both inherited from the Transportation Network parent object, each acquiring the display attributes and the ShowMe and HideMe methods. In addition, railroad network inherits the Select Segment, Valid Segment, and Unselect Segment methods while highway network inherits the Generate Impact Segment and the HighlightImpacts methods. The Grain Elevators and Fertilizer Centers are inherited from the Intermodal Facility Location object. Both of these objects inherit their railroad locations and the ShowMe and HideMe methods. Figure 7 shows examples of object inheritance in the Railroad Branchline Abandonment Use Case.

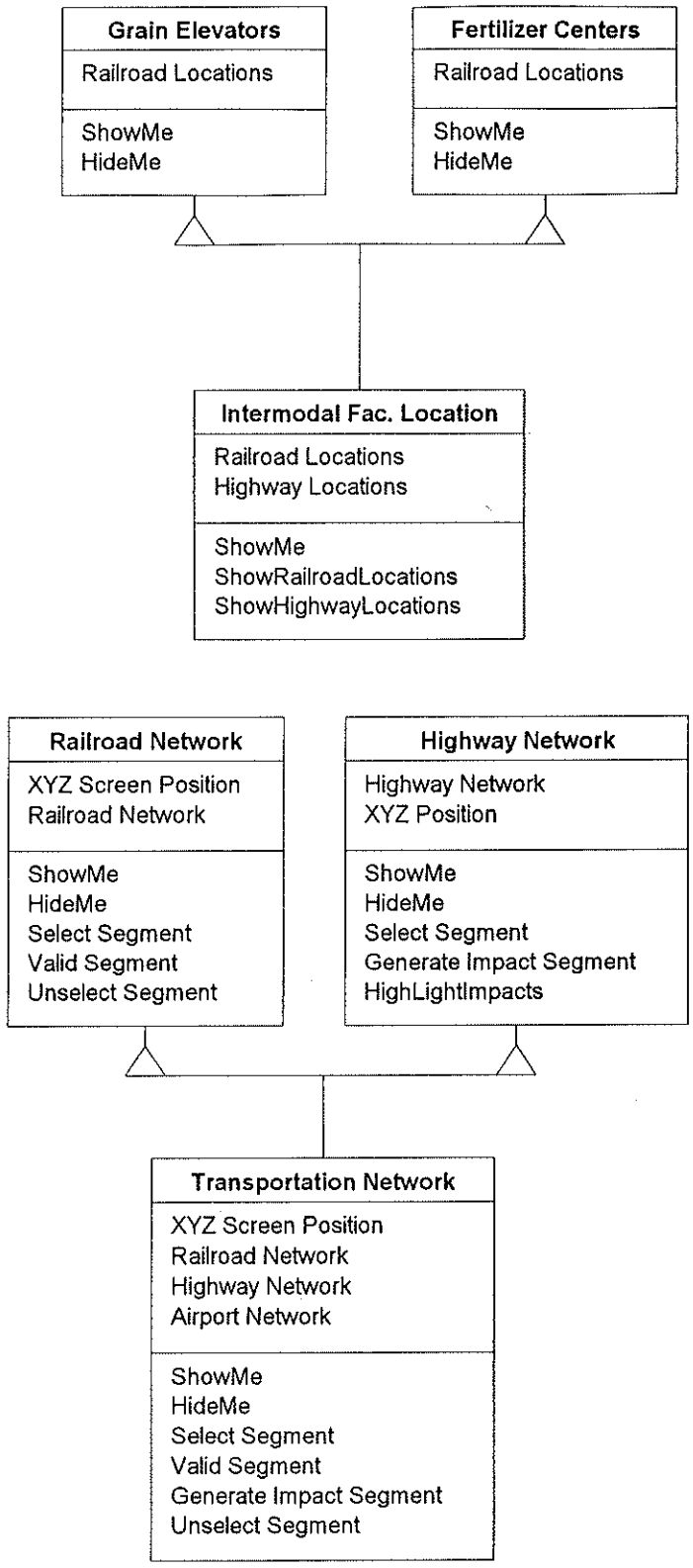
3.3.4 Encapsulation in the Railroad Branchline Abandonment Use Case

In object-oriented technology, objects contain or encapsulate the data and the data operations of an entity in the system. Encapsulation separates the external aspects of an object, which are accessible to other objects, from the implementation details of the object, which are hidden from other objects. It is the employment of information hiding, the process of hiding an object's internal structure and implementation detail from the system. An object may be accessed or used through its interface or specification, but the actual internal object detail is hidden.

In the *Railroad Branchline Abandonment* use case, the high level objects are declarative in nature and shield the analyst from any implementation detail. For example,

Figure 7. Examples of Object Inheritance in the Railroad Branchline Abandonment Use Case Object Specification

(Next Page)



Example of Object Inheritance in the Railroad Branchline Abandonment Use Case Objcect Specification

the HIA Start Window object has several attributes and methods, but no implementation detail is described as part of the object. When HIA Start Window is used in the system, only the public attributes and methods are known, not anything about how these attributes and methods are implemented. If the RBA Icon object wants the HIA Start Window object to perform the Show/Hide Transportation Networks method, it sends a message to the HIA Start Window object requesting that it be done. The RBA Icon object does not need to know how the HIA Start Window object implements the method, only that the method is available. This hides or encapsulates the implementation details of the HIA Start Window object. All other objects in the system use encapsulation.

In object-oriented analysis, the analyst need only know the attributes and methods of an object. Encapsulation allows for a high level of system abstraction and eases the conceptual burden on the analyst by hiding the implementation details of an object.

3.3.5 Reuse of Railroad Branchline Abandonment Objects

One of the major benefits of object-oriented technology is the reuse of objects (Graham, 1994). This section describes the reuse of the objects specified for *Railroad Branchline Abandonment* by the other use cases. The Grain Elevator's object will be reused by three other use cases: 1) *Intermodal Facility Efficiency,* 2) *Estimate Highway Investment Costs,* and 3) *Grain Elevator Modal Split.* The Fertilize Center's object will be reused by the *Intermodal Facility Efficiency*, *Estimate Highway Investment Costs,* and *Fertilizer Distribution Facility Location* objects. The Railroad Network object will be reused by the *Railroad Branchline Rehabilitation Evaluation* use case while the Highway Network will be reused by the *Estimate Highway Investment Costs, Fertilizer Distribution*

Facility Location, and *Railroad Branchline Rehabilitation Evaluation* use cases. The HIA Start Window object will be reused by the *Railroad Branchline Rehabilitation Evaluation* use case. In addition, the generic display and dialogue objects not illustrated in the diagram, but used in *Railroad Branchline Abandonment,* will be reused throughout all the other use cases. Additional reuse of these objects may also be identified during the design phase of IMS development. These objects would also be available for reuse in the development of other transportation management systems, starting the formation of a library of reusable transportation objects. Figure 8 shows the high level reuse of objects in the Railroad Branchline Abandonment Use Case.

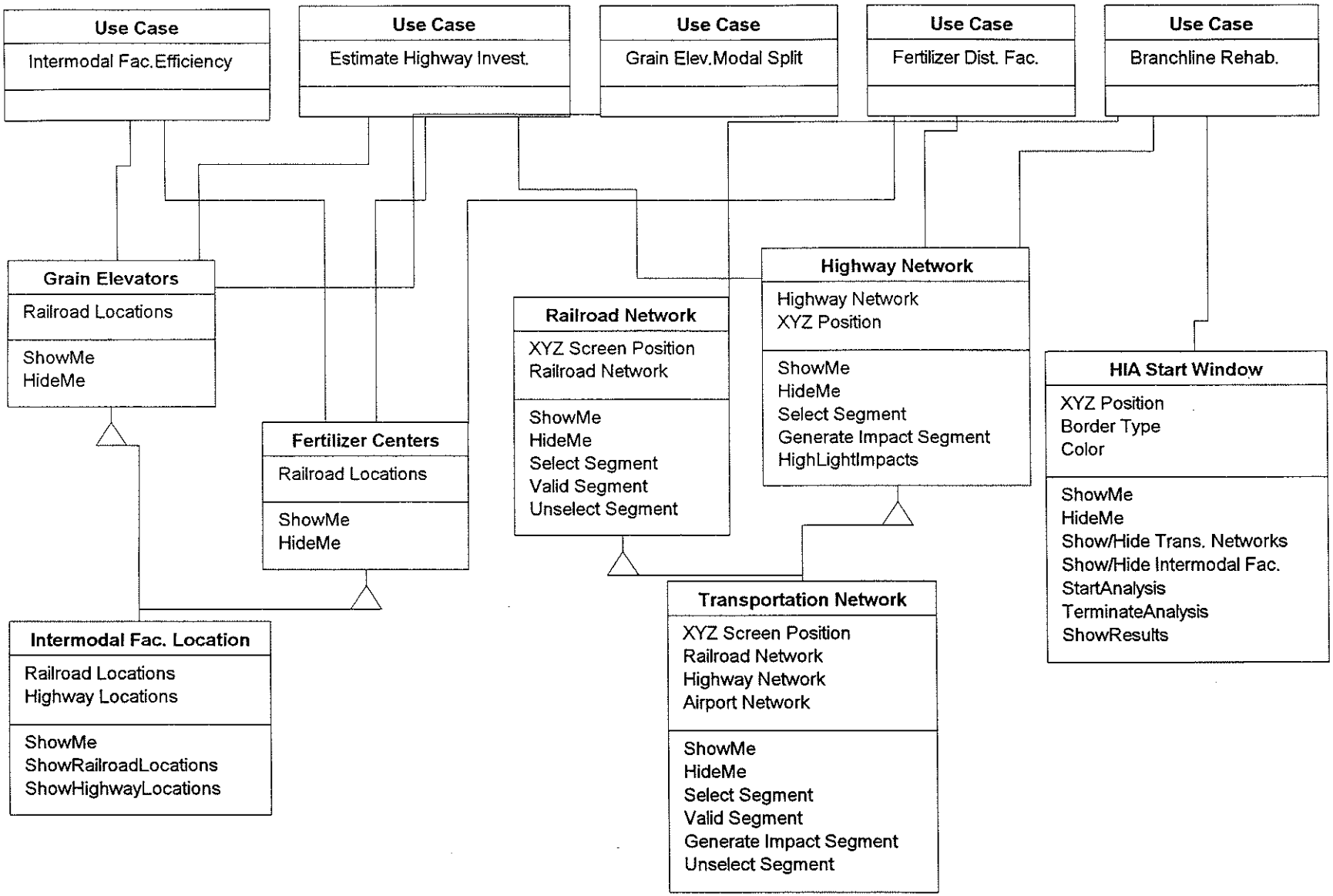
3.4 IMS User Centered Object-oriented Analysis vs Data-driven Analysis

This section compares and contrasts the IMS user centered object-oriented approach developed in this chapter with a more traditional design approach, a data-driven approach. The purpose of this comparison is to highlight the advantages a user centered object-oriented approach offers for IMS development. The comparison is done by considering two important, generic ideas in system design: 1) understanding the user and 2) managing a complex system.

3.4.1 Understanding the User

Understanding the user and how the user will utilize the system is fundamental to successful system design. In the approach adopted by this paper, the use case methodology of Jacobson, the focus of the analysis centers and remains centered on how the user will use the system. The development of use cases dictates that the terms of system development will be on how the system is used. Use cases drive and control system analysis and, when

Figure 8. High Level IMS Reuse of the Railroad Branchline Abandonment Use Case Object Specification
(Next Page)


Use Case
Intermodal Fac.Efficiency
Use Case
Estimate Highway Invest.
Use Case
Grain Elev.Modal Split
Use Case
Fertilizer Dist. Fac.
Use Case
Branchline Rehab.
Grain Elevators
Railroad Locations
ShowMe
HideMe
Fertilizer Centers
Railroad Locations
ShowMe
HideMe
Railroad Network
XYZ Screen Position
Railroad Network
ShowMe
HideMe
Select Segment
Valid Segment
Unselect Segment
Highway Network
Highway Network
XYZ Position
ShowMe
HideMe
Select Segment
Generate Impact Segment
HighLightImpacts
HIA Start Window
XYZ Position
Border Type
Color
ShowMe
HideMe
Show/Hide Trans. Networks
Show/Hide Intermodal Fac.
StartAnalysis
TerminateAnalysis
ShowResults
Intermodal Fac. Location
Railroad Locations
Highway Locations
ShowMe
ShowRailroadLocations
ShowHighwayLocations
Transportation Network
XYZ Screen Position
Railroad Network
Highway Network
Airport Network
ShowMe
HideMe
Select Segment
Valid Segment
Generate Impact Segment
Unselect Segment

properly used, do not allow developer perceptions of the system to deviate from the user's model of the system.

In contrast, a developer's data-driven approach would center on the functionality surrounding data as it relates to a system specified function. This type of analysis concentrates on the development of the data and the data operations to define the system, not the user's model of the system. A data-driven approach concerns itself more with data than with the user, and the behavior of the user is secondary. Because of this emphasis during system analysis and design, this approach does not understand the user as well as a user centered approach.

The railroad branchline abandonment analysis and design scenario starts when the user says, "I want to be able to do railroad branchline abandonment." The data-driven approach asks what data are needed for a railroad branchline abandonment. It follows by asking what must be done to the data to perform a railroad branchline abandonment. This system is designed around the data. Design decisions are based upon making the system do data manipulations. The user centered object-oriented approach asks how does the user do railroad branchline abandonment, and what do we create to match how the user does railroad branchline abandonment. This system is designed around the user. Design decisions are based upon making the system act similarly to how the user acts and models a railroad branchline abandonment.

For example, in the *Railroad Branchline Abandonment* use case, the user centered, object-oriented methodology approached this scenario by determining that the user wanted to do railroad branchline abandonment and finding out how the user would do it in terms of

interaction with the IMS. A data-driven approach would center on what must be done to the railroad branchline abandonment data to support the functionality requested by the user. In the first case, this would lead to conceptualizing system entities that have a counterpart in the user's model of the system. Examples of this are the PMS object, the transportation networks, and the intermodal facilities objects. The second case would lead to conceptualizing system entities that are functionality centered on the railroad branchline abandonment data. Examples of this would be developing data flow diagrams depicting how the data would flow through the system as it performs a railroad branchline abandonment.

More specifically, continuing with the *Railroad Branchline Abandonment* use case example, the user centered object-oriented approach develops entities based upon the user tasks involved in completing this activity. The HIA Start Window shows what the user wants to see while doing this activity while the other objects model how the user described this activity. In a data-driven approach, the developer forms an understanding of the system from what data manipulations are required to meet user expectations. The one approach remains focused on the user through the use case model while the other approach is centered around the data. At the end of the analysis process, one would understand the system the way the user would use the system while the other would understand the system by what data functionality the system must do to meet user expectations. The first remains centered on the user, the second on the use of data. By remaining centered on the user, the user centered object-oriented approach will more closely follow the user's model of the system and will lead to a system design based on a more complete understanding of the user.

3.4.2 Managing a Complex System

A major challenge to developing large, complex systems is the management of complexity. The North Dakota IMS will use a significant number of databases and may interface with several other transportation management systems. Table 4 outlines potential IMS databases.

Table 4. Potential Intermodal Management System Databases.

IMS Used Database	Number of Fields	Number of Records
Coal	3	14
Airports	17	100
Transit	14	53
Fertilizer	15	228
Grain Elevators	50	425
Motor Carrier	25	106
Sugar Beet	4	30
Default Highway Data I	12	26
Default Highway Data II	7	16
Elevator Routes	11	1275
Rail Access Route	3	100
North Dakota Waybill	160	15000/year
North Dakota PMS	27	1400
North Dakota Cross.	83	5000

Included in the table are the basic seven IMS databases introduced in Chapter 1 including the North Dakota PMS used in the railroad branchline abandonment use case, two Default Highway Databases used in the railroad branchline abandonment use case, the North

Dakota Waybill used for tracking commodity shipments, the North Dakota Railroad Crossings files used for assessing transportation network crossings, and two route files. This list is not exhaustive, but illustrates the scale and complexity of the system data. The IMS will also access the GIS and its collection of spatial data, Oracle tables, and Info tables. In addition, the 40 or more major transactions expected of the system, as indicated in one of the preliminary surveys, along with the potential interaction with other transportation management systems, illuminate the complexity of the North Dakota IMS.

Object-oriented analysis and design manages complexity in several ways. One is the use of abstraction. Abstraction is the elimination of the need to consider the non-essential details of an entity. The object concept used in object-oriented technology is an abstraction of an entity through the use of encapsulation and information hiding. Encapsulation draws together the essential characteristics of an entity, its attributes and behaviors, into an abstraction, the object. Information hiding conceals the internal implementation details of an object. Thus, the analyst is removed from the internal complexity of an entity by using the object concept. In the railroad branchline abandonment use case, several of these databases are used, but from a conceptual point of view, the use of objects hides the details of implementing code for utilizing the databases. The PMS database is the PMS object in the railroad branchline abandonment use case. To the object-oriented analyst, it gets and validates PMS data. The PMS database is a different concept in a data-driven approach. It is a collection of records having a series of related fields that need to be accessed and need to be validated. The abstraction offered by object-oriented technology provides the analyst a simpler concept with which to build a system. As pointed out by Booch (1991), an

object-oriented problem domain decomposition results in fewer conceptual entities when compared to an algorithmic decomposition common to data-driven approaches. In the algorithmic decomposition, each operation or method of an object in the object-oriented decomposition would result in an entity in the decomposition. In the object-oriented decomposition, the objects would encapsulate various algorithmic decomposition entities into one abstraction, an object. The fewer entities in the object-oriented approach relieve the analyst of the additional conceptual burden of more entities in the model.

The scale of the data in the system illustrates another advantage an object-oriented system would have in managing the complexity of the system as the data changes and evolves. In an object-oriented system, the object inheritance structure, or hierarchy, allows the user to easily make changes to all the objects in the system associated with any changes to the data structure or related calculation or interface changes. One change at the parent level, incorporating any change or evolution of the data structure, is propagated throughout its entire object hierarchy in the system. In contrast, a data-driven approach must search the entire system to find where the data changes will have an effect and make those changes at each location that is associated with the changed data. This can be a more onerous process with a greater likelihood of error because of all the individual localized changes possible.

In particular, the user interface of a data-driven system would require more effort in responding to the evolution of the data structure. The user interface is more directly related to the data structure in a data-driven system whereas the interface objects in an object-oriented system are more independent of the data structure.

The reusability of objects in an object-oriented system also helps to manage the complexity of the data. If the IMS needs to access and use data from a particular source, it may use an existing object for that task, making whatever modifications are necessary for the new object. The new modifications may incorporate attributes or methods from other existing objects, inheriting them and forming a new, compound object. The data-driven approach would need to develop a new, separate code module to perform the same activity. In this situation, the object-oriented system again offers the analyst the conceptual luxury of dealing with a high level of abstraction with the use of objects rather than thinking about the level of detail necessary in developing a new code module.

Other object-oriented advantages in managing complexity include the localization of information in distinct objects in contrast to using variables of various scope in a data-driven approach. Moreover, the reuse of objects also provides a readily available means of extending the system and managing the possible complexity involved in any system extension.

3.5 Railroad Branchline Abandonment Interface Description

The interface description segment of the requirements model is created to provide a mechanism for assuring that requirements modeling is consistent with the *transportation planner*'s logical view of the system. The interface description will involve *transportation planner* and will simulate the use cases to gain feedback to modify the system to meet user expectations. Jacobson's methodology is augmented in this paper with a discussion of user centered interface design. This is done in Chapter 4 and includes user interface design decisions and guidelines to make this part of the analysis process more user centered.

CHAPTER 4. IMS USER CENTERED PROTOTYPE

4.1 Introduction

This chapter supplements Jacobson's interface description by utilizing user centered design to support and improve design decisions for a user centered IMS prototype. Prototyping is the technique of creating a prototype of the main features of a system and demonstrating these features to the user early in the design process as part of an effort to keep the design centered on the users. User centered design approaches system design from the perspective of the potential user, developing user centered design information that may be utilized during the design process. The user centered design environment and information developed in this section will be used in making design decisions about the development of the IMS prototype.

The chapter starts with a section providing some background on user centered design. Presented first in this section are several approaches to user centered design followed by subsections on 1) interface design, 2) information systems, and 3) IMS activities. Interface design is included because it is an issue vital to any user centered discussion. The user interface, to the user, is the system itself, and great care must be utilized in designing a user interface. The discussion on information systems is presented to provide context for the IMS which is, in many respects, an information system. The related IMS activities section describes meetings held with NDDOT officials, Intermodal Management conferences attended, and evaluations of other state's efforts. The related IMS activities section also includes additional information about potential users and the user context.

4.2 User Centered Design Context

Norman and Draper (1986) consider several approaches or design methodologies to the issue of user centered system design as applied to computer systems developed for humans. Each of these design methodologies offers a different perspective for gaining an understanding of the user and the user's environment. This understanding is fundamental to user centered design. The first approach is a quantitative one dealing with specific measurements of the human and the system while the next two methods take more of a qualitative, subjective approach.

The first approach begins by considering the person or user and the human information processing structure, utilizing this knowledge as a framework for designing a human-computer interface. This approach was used by Card, Moran, and Newell (1983) in the development of an applied information-processing psychology of human-computer interfaces. Card, Moran, and Newell employed cognitive theory and human information processing performance measures to develop user performance models predicting the performance of a human-computer system. From these models, they derive ten user centered system design principles of which two can be applied to a system prototype:

1. Early in the system design process, consider the psychology of the user and the design of the user interface.
3. Specify the user population.

The remaining eight user centered system design principles address issues appearing later in the design process.

A second approach examines the user's subjective experience and considers how it may be enhanced by human-computer interaction. It basically asks what the experience is like for the user. This approach contemplates the idea of "direct engagement" whereby the user becomes an active participant in the computer application, much like someone watching a movie may feel that they are having a first person experience rather than being a spectator (Laurel, 1990). The challenge is to have the user believe they are a participant in the computer's activity, not someone removed merely directing the computer system. Laurel is an advocate of this approach, asserting that the cognitive and emotional aspects of a user and the user's experience must be central to designing for the user. She maintains that human-computer interface design is an art and will remain so. Laurel writes that an interface "must have qualities which enable a person to become engaged, rationally and emotionally, in its unique context."

Heckel (1991) promotes this approach and states that "writing friendly software is a communications task, and to do it effectively you must apply the techniques of effective communication; techniques that are little different from those developed by writers, filmmakers - virtually anyone who has attempted to communicate an idea over the past decades, centuries, even in some cases millennia" (Heckel, 1991). He adds that user centered software design is a communications craft best learned from writers, painters, and filmmakers whose perspective is one of having an effect on the individual's emotion and cognition.

The third approach centers on the social context of the computer and the computer's use as a tool for completing specific user tasks (Brown, 1986). This approach evaluates the

user's work tasks, the social and work environment in which the users operate, and the nature of the process combining the two. The design process attempts to understand this and to develop a system that integrates into the user's social and work environments. Brown considers this approach and argues that the best way to influence social change is in the design of computer systems. He uses the example of distribution lists on electronic mail systems as a way to impact the behavior of organizations. This approach uses the broadest context of those discussed.

4.2.1 Interface Design Background

The user interface is an important part of any computerized system. It is essential that any system development thoroughly consider the design and implementation of the user interface, particularly from the perspective of the user. To the user, the user interface is the system. The following discussion presents two views of interface design.

Heckel (1991) asserts that computer software is undergoing an evolution similar to that of filmmaking, from what is presented to how it is presented. He says that writing software is a communications task and the user interface is the medium of communication. Heckel's approach emphasizes high level visual thinking and communication, and his design principles emphasize knowing the user, communicating with the user, and keeping the user engaged with the program. He continually stresses the idea of communication, particularly visual communication, and offers this advice to the interface designer:

> You must learn to think like a writer: in terms of images, clarity, and impact. I find it fascinating to see how members of different professions think. Lawyers, businessmen, accountants, engineers, programmers, and writers all think differently.

As a designer of user interfaces, it is crucially important that you observe your audiences think, so you can design your user interfaces with their thought processes in mind. You must realize that you are communicating with someone.

Marcus (1992) approaches interface design from a more detailed human-computer interaction (HCI) perspective. While succinctly describing a comprehensive set of high level interface design goals, he elaborates most fully on the appearance and interaction attributes of a graphical screen display. He presents detailed design guidelines for developing effective spatial displays that can provide users with helpful visual communication constructs.

One of Marcus' major user interface concerns is the effective use of the space on a screen display. Marcus (1992) calls proportion and grids the invisible keys to a successful screen display and defines these concepts as a set of horizontal and vertical lines that divide the visual field into units that have visual and conceptual integrity. The most effective visual communication requires that the user perceive a pleasing screen proportion, and careful consideration should be given to designing the screen layout. A primary objective should be to develop informational display areas on the screen display that maintain an adequate spatial relationship with the current screen layout.

Marcus (1992) also addresses the issue of the use of color and calls color the most sophisticated and complex of the visible language components. He lists several advantages to using color, including increasing the appeal, believability, memorability, and comprehensibility of the interface. Disadvantages included the existence of color deficient users (8 percent of Caucasian males) and the possibility of visual fatigue or visual confusion from strong or complex color phenomena. His color guidelines indicate that red and green

should be used in the center of the visual field, not in the periphery as the edges of the retina are not particularly sensitive to red and green. He further states that black, white, yellow, and blue should be used in the periphery of the visual field as the retina remains sensitive to these colors in the periphery. The following summarizes Marcus's other recommendations concerning the use of color.

1. Use blue for large areas, not for text type, thin lines, or small shapes.
2. Use spectral order in color coding (i.e., red, orange, yellow, green, blue, indigo, violet). Tests have shown that viewers see a spectral order as a natural order.
3. Use a color area that exhibits a minimum shift in color and/or size if the color area changes in size. As they decrease in size, color areas appear to change their value and chroma. Consequently, color interaction with the background fields becomes more pronounced.
4. Use familiar consistent color codings with appropriate references.
5. Use color for quantitative coding as well as for qualitative coding. The degree of color change can be linked to some magnitude change in the displayed process or event.
6. Use high-value, high-chroma colors to attract attention. The use of bright colors for danger signals, attention getters, reminders, and cursors is entirely appropriate.

7. Command and control colors in menus should not be used for information coding within a work area, unless a specific connection is intended.

Another of Marcus's major graphical display concerns is typography. Marcus (1992) points out that studies have shown that both legibility and readability can be significantly improved through careful selection of the type and layout of the material. Rehe (1974) found that words set in all capitals use up 30 percent more space for variable-width letters and retard reading speed by 12 percent. Marcus further notes that word shapes are crucial for efficient reading and capital letters with regular height reduce the variability of the word shapes presented by uppercase and lowercase letters together.

Marcus (1992) has developed an extensive set of general HCI design principles derived from his professional practice and adapted from those set forth in Baecker and Marcus (1990). This section concludes with these specific, low-level principles to interface design.

1. Legibility: Design the individual characters and symbols of the textual and graphical vocabulary for an Human-Computer Interface (HCI) visualization so that they are rapidly and reliably identifiable and recognizable.
2. Readability: Design the textual components of an HCI so that they are easy to interpret and understand.
3. Clarity: Design all non-textual components of an HCI so that their semantics are as unambiguous as possible.

4. Simplicity: Include in an HCI only those elements that communicate something important. Try to be as unobtrusive as possible.

5. Economy: Maximize the effectiveness of a minimal set of techniques or cues.

6. Consistency: Observe the same conventions and rules for all elements of an HCI. Be consistent from HCI to HCI. Deviate from current conventions only when there is a clear benefit to be gained.

7. Relationships: Use visible language elements to show relationships among those elements of an HCI that need to be linked and to show lack of relationship among those that should not be linked.

8. Distinctiveness: Use visible language elements to distinguish important properties of essential components of an HCI.

9. Emphasis: Use visible language to emphasize the most salient features of an HCI.

10. Focus and Navigability: Use visible language to position the initial attention of a reader or viewer to the HCI or one of its components, to direct attention, and to assist in navigating around the material.

11. Screen Characteristics: Select visualization techniques appropriate to the output display technology.

12. Screen Composition and Layout: Clarify the HCI structure and emphasize important relationships by carefully organizing the screen composition through application of a standardized screen layout,

through the use of implicit grids, through the application of explicit and implicit rules, and through the inclusion of appropriate white space.

13. Typographic Vocabulary: Choose a small number of appropriate typefaces of suitable legibility, clarity, and distinctiveness, applying them thoughtfully to encode and distinguish various kinds of HCI tokens. Within each typeface, select a set of enhanced letter forms, punctuation marks, and symbols with which to present the text effectively.

14. Typesetting: To enhance the readability of the program, to emphasize what is most salient, and to achieve appropriate focus, adjust the typesetting parameters of text point size, heading size and usage, word spacing, paragraph indentation, and line spacing.

15. Symbolism: Integrate appropriate symbols and diagrammatic elements to bring out and highlight essential HCI structure.

16. Color and Texture: Use appropriate highlighting and low lighting techniques, such as one or two levels of gray tone as backgrounds, if they assist in meaningful semantic distinctions.

17. Metatext: Augment the HCI elements of text and graphics with automatically generated supplementary text and graphics (metadata, metatext, or metagraphics) that enhance the comprehensibility and usability of the original elements.

18. Simultaneous Views: Provide the ability to view simultaneously a major "focus" plus additional information on a single screen.

19. Access Facilitation: Provide links and cross-references among these views, and facilitate navigation through the contents using these mechanisms.

20. Regularity and Similarity: Maximize the similarity of major kinds of HCI metatext, e.g., on-line manuals, guides, and quick references or indices. Maximize the regularity of the location and appearance of individual items of metatext.

4.2.2 Information Systems Background

An information system may be defined as a structure for providing information to support the operations, management, and decision-making functions of an organization (Davis, 1974). The IMS is an information system with built-in modeling, providing information about various intermodal transportation questions. This section provides a broader context for considering the IMS as a user centered system within a larger information-based organization.

Several approaches for information system development are reported in Avison and Wood-Harper (1990). These are individual methods concerned primarily with technical problems, strategic implications for organizations, cost reduction, or changes in the nature and content of the work done by the organization. Avison and Wood-Harper support Multiview, an approach that draws on research from several other related areas. Multiview draws on the analysis of human activity systems, socio-technical systems, data analysis, and

structured analysis to account for the various points of view of the users involved in a computerized information system. Avison and Wood-Harper formulate five questions that Multiview seeks to answer in developing a system specific, practical framework for information system development.

1. How is the information system supposed to further the aims of the organization using it?
2. How can it be fitted into the working lives of the people in the organization who are going to use it?
3. How can the individuals concerned best relate to the computer in terms of operating it and using the output from it?
4. What information processing function is the system to perform?
5. What is the technical specification of a system that will come close enough to doing the things that you have written down in the answers to the other four questions?

Kirk (1993) asserts that the most important part of any information system is the people. People recognize the need for a system, design and implement the system, and use the system. Any information system originates from people and is developed to assist people in accomplishing specified, usually work-related tasks. Therefore, the focus for information system development must be on the most important characteristic of the system, the people. Kirk's approach more closely approximates a user centered design paradigm with its emphasis on the people using the system.

4.2.3 Related IMS Activities

A series of meetings was held with officials from the NDDOT to discuss and consider development of the IMS. These meetings addressed several issues including the potential use of an IMS and emphasized three points in that regard. The first two points were that the IMS must be quickly seen as being useful and easy to use. This places added importance on designing an effective user interface to capture the user's interest, much like the engagement discussed by Laurel (1990), and on understanding the user to make the system easy to use. The third point was that the NDDOT, like many other states, does not think in intermodal terms. North Dakota, again as many other states, remains primarily focused on the state's highway system and in such a way that relates mainly to highway engineering and highway developmental projects. The IMS, to be successful, must draw on that interest and present the IMS within that context. Doing so would draw upon the work of Brown (1986), emphasizing the social and work environment of the system, and Avision and Wood-Harper (1990) who cast information systems in terms of how it will further the aims of the organization using it.

To gain some insight into other states' efforts, two intermodal transportation conferences also were attended, and evaluations of California and New Mexico preliminary design documents were done (Barton-Aschman, 1993; Carter, 1993).

4.3 IMS Computer Use Survey

A survey was developed to gather basic, preliminary information about the user. The Intermodal Management System Computer Use survey was sent to potential IMS users at the North Dakota Department of Transportation. The survey was used to facilitate the

translation of the design principles into design decisions for a user centered IMS prototype. The four survey categories were 1) Computer Use, 2) Graphical User Interfaces (GUI), 3) Information Use, and 4) Intermodal Transportation. A general comment section was included at the end of the survey. The survey, while not exhaustive, provides a starting point for capturing user characteristics necessary for implementing user centered design and illustrates the process of user centered design.

These categories were selected to determine the users' basic computer skills, their experience with graphical user interfaces, the kind of information they used, and their experience and perception of intermodal transportation. The purpose of the first two sections, following Card's (1983) design principles 1 and 3 and Laurel's (1990) interest in the user's experience, is to gain a sense of the users' computer experience. While the questions in these first two sections are requesting basic information, these sections are also useful for detecting unusual aspects of the users' computer experience. Section 3 draws on Avison and Wood (1990) as well as Kirk (1973) and was used to understand how the users are utilizing information in their work. The fourth category also draws on Avison and Wood and sought to elicit the current and future perspectives on intermodal transportation. The last survey category was a comment section.

4.3.1 Survey Participants

The planning division of the NDDOT was selected to be surveyed because that department was identified by the NDDOT IMS coordinator as having the most active users. These users included the planning professionals, particularly the transportation planners. This department also has management personnel and various technical professionals as

potential users. This broadened the category of users and offered a wider look at potential users.

4.3.2 Survey Participation Rate

The Intermodal Management System Computer Use survey was sent to 37 potential IMS users. Thirty responses were received for a return ratio of 81 percent.

4.3.3 SECTION I. Computer Use

The computer use section of the survey queried users about the type of computer experience and computer literacy they had. Questions were asked about the hardware and software that users were familiar with as well as how long they had been using computers (Figure 9).

This part of the survey was used to gain a rudimentary understanding of the users' computer experience. As pointed out by Card (1983) and Laurel (1990), it is important to know the users as this allows the designer to build upon the experience of the user. In this regard, questions were asked concerning the type of computer used, how long a computer had been used, how much time is spent on

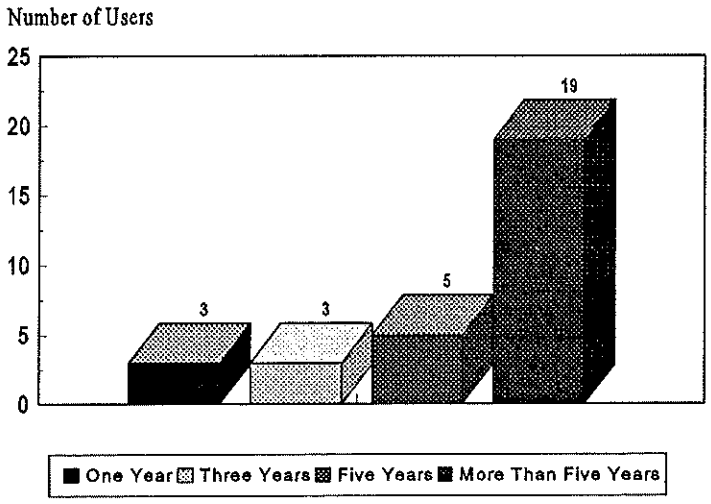


Figure 9. Years of Computer Experience

the computer, what operating system is used, and other computer demographic questions (Figure 10).

The respondents all used a PC with only 2 (7 percent) also using a workstation and 4 (14 percent) a mainframe. This response indicates that a PC based system will be a familiar hardware platform as opposed to any significant split between mainframe and PC users. A large majority of users, 80 percent, had 5 or more years experience with computer with only 3 users being novices (1 year or less).

The operating systems used by the respondents were concentrated on two, DOS and MicroSoft Windows. DOS was the most used operating system with 22 (76 percent) respondents using it while MicroSoft Windows was used by almost half (48 percent) of the users. However, discussion subsequent to the survey indicates that almost all of the NDDOT planning division has used MicroSoft Windows.

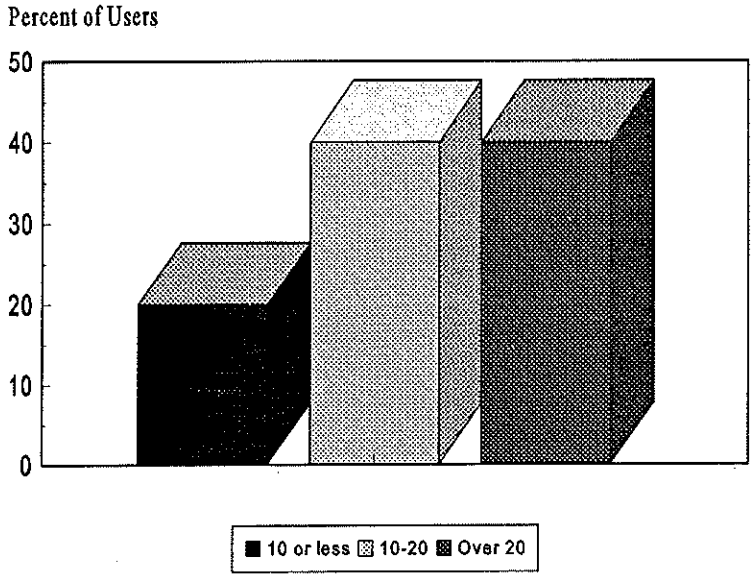


Figure 10. Computer Hours per Week

Most (82 percent) respondents used their computers for 10 or more hours a week with 41 percent spending over 20 hours per week. This indicates that a significant portion of most users' work week is spent with their computers.

Three common application programs were chosen as the most used or most familiar software programs: 1) word processor with 23 users (77 percent), 2) database with 22 users (72 percent), and 3) spreadsheet with 18 users (59 percent) (Figure 11). Of particular note are the 10 users (33 percent) who use a CAD (Computer Aided Design) program, a particularly high percentage for a typical user sample. CAD programs are visually oriented programs with a spatial dimension reflecting the use of spatial and design information in the planning division. The use (43 percent) of graphics programs reinforces the spatial and graphical orientation of the planning division's computer use. The high percentage of database and spreadsheet users indicate experience with numeric information and relationships among data elements and types. Only one respondent was familiar with decision support systems.

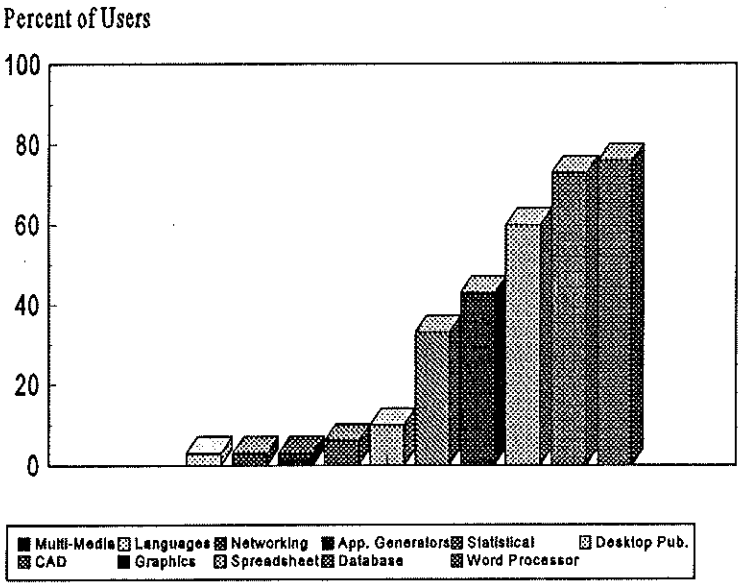


Figure 11. Software Programs Utilized

Three of the potential users surveyed develop computer programs or information systems. This question

was asked to test the depth of understanding about the computer science involved in an information system with the hypothesis being that the more depth the user displayed, the more he would understand the underlying model of an information system. With only three responses, the survey indicates that the depth of computer science knowledge is not significant.

4.3.4 SECTION II. Graphical User Interfaces (GUI)

This section focuses on the user's experience and knowledge of Graphical User Interfaces (GUI) and, in conjunction with Section 4.3.3, will be used to fashion the basis of selecting an early prototype hardware and software developmental platform. Questions were asked to determine what experience users had with GUIs and what their perceptions were regarding GUIs. Discussion subsequent to the survey indicates that experience with GUIs had grown and expanded.

More than one-half (55 percent) of the users have used a graphical user interface with one-third (33 percent) of the respondents using MicroSoft Windows. MicroSoft Windows, by a large margin (33 percent versus 3 percent), was the most used graphical user interface.

A majority of respondents, 53 percent, used a graphical user interface in their work. Respondents, as a group, demonstrated familiarity with graphical user interfaces.

Of those respondents using graphical user interfaces, almost all (94 percent) found them to be somewhat or a lot helpful. Seventy-five percent considered GUI "a lot" helpful, and only one respondent thought GUIs were only "a little" helpful.

4.3.5 SECTION III. Information Use

The information use section intended to learn what kind of information was utilized in the users' work environment. This was done to build on this experience in developing a prototype incorporating the existing patterns of utilizing information.

Eighty-seven percent of the users reported that they use raw data in their work or information environment (Figure 12). This suggests that users are analyzing and condensing original data into more refined information for further analysis. Sixty-three percent of the users indicated that they use written reports and maps in their work. Most respondents reported using some type of graphic information in their work whether it is maps (63 percent), graphs (60 percent), or charts (43 percent). The high use of graphic information indicates familiarity and experience with visual communication. Almost one-half (47 percent) also used statistics at work.

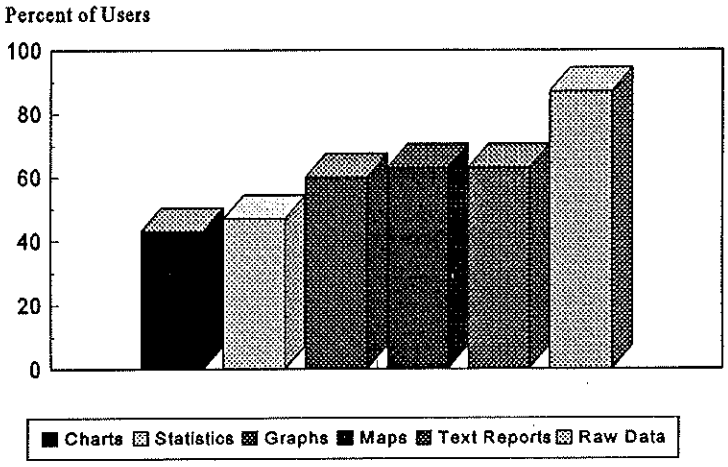


Figure 12. Types of User Information

A minority (40 percent) of the respondents use a computerized information system like the NDDOT's Pavement Management System (PMS). This question was asked to test users' experience with transportation management systems. It appears that a sizable

minority has experience with other management systems making an introduction to the IMS easier.

Twenty-one different responses were received describing the type of information the users found most useful in their work. Of the ways that user information is utilized, analysis and presentation reports were the most frequent ways (73 percent). Decision making and writing reports were the second-largest (66 percent) ways, while making charts was third (57 percent) although still used by a majority of users.

4.3.6 SECTION IV. Intermodal Transportation

This section queried potential IMS users about their experience and perceptions of intermodal transportation (Figure 13). Of particular interest was determining the attitude toward the future importance of intermodal transportation.

Only 1 respondent indicated a lot of experience with intermodal transportation while 7 (23 percent) respondents said they had no experience at all. Over 60 percent of the users replied that they had a little or some experience with intermodal transportation. The experience base of intermodal transportation, while not extensive, is notable.

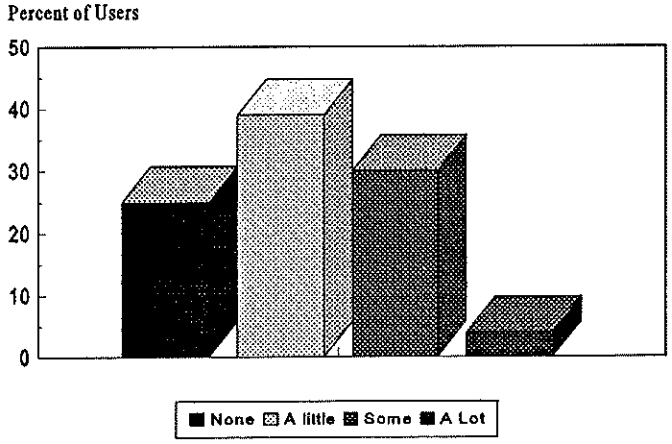


Figure 13. Intermodal Transportation Experience

Fifty-seven percent of the users indicated that they need to consider intermodal transportation in

their job "somewhat" or "a lot." Only 3 respondents said they do not have to consider intermodal transportation. All of the responses to a question asking if intermodal transportation affected their department indicated that intermodal transportation affected their department "some" or "a lot." All users responding to the question whether intermodal transportation would become more or less important indicated that it would become more important.

4.3.7 SECTION V. Comments

Seven survey respondents included comments on their survey. Four of the comments concerned intermodal transportation and included two comments that either said intermodal transportation needs to be defined or that intermodal transportation information is often unavailable. One respondent indicated that intermodal transportation would become more important because of the requirements of ISTEA. The final intermodal comment concerned a supervisor/administrator who said he had directed urban highway improvement projects that got into intermodal issues.

The remaining comments were either a synopsis of that individual's work tasks or a lengthy comment on how good he considers North Dakota's transportation system.

4.3.8 Survey Discussion

The survey shows that the potential users of the IMS are knowledgeable, long-time PC users who are familiar with working with numerical analysis and report development. Many are familiar with CAD, database, and statistical programs. Most of the potential users utilized mapping, graphing, and other visual communication techniques in their work while a sizable minority had used another transportation management system. All the potential

users thought intermodal transportation was going to be more important in the future, and a majority had a little or some experience with intermodal transportation. The basic finding of the survey is that, from a computer use, information use and intermodal transportation perspective, the IMS should be a PC based system, visually oriented particularly with respect to displaying transportation networks and will be introduced into an environment considering intermodal transportation important.

4.4 IMS Prototype Design Decisions

Sections 4.2 and 4.3 provide design principles and basic user information that are used to make design decisions about the IMS prototype. Section 4.2 presented various design approaches and design principles for making a user centered system. Section 4.3 discovered basic user information that can be incorporated into an IMS prototype.

The railroad branchline abandonment use case is used to illustrate the application of these design principles and design information. Design decisions about the visual objects in that use case are presented with the design background source included in brackets.

1. RBA Icon: This icon will be the same size as those used in WordPerfect and will be positioned in a toolbar identical to WordPerfect's at the top of the screen display {Marcus (1992), Laurel (1990), use case surveys}. It will have an image of a railroad branchline partially overlaid with a question mark {Laurel (1991), Heckel (1991)}. The icon will be colored a light gray identical to the gray of the WordPerfect icons, and the icon image will be colored black for the branchline with an unobstrusive light maroon for the question mark {Heckel, Marcus, use case surveys}. The light maroon will be

the color identified with the railroad branchline abandonment process and is selected to attend to the center of the visual field while not being too bright to signal any need for immediate attention {Marcus}. The icon will be displayed in a section of the tollbar reserved for use cases whose major intermodal concept is that of a railroad {Avison and Wood-Harper (1990)}.

2. HIA Start Window: This window will be a full screen display underneath the top toolbar. It will incorporate the light maroon coloring of the railroad branchline process at the periphery of the window {Marcus (1992), Laurel (1990)}. The window will display the transportation networks and intermodal facilities and will have a blue background for these objects {Marcus}. An optional background color is the background color used by the CAD program identified in the computer use survey {Laurel, computer use survey}

3. Railroad Network: This object will use white to display the railroad network and will use reasonably heavy lines {Marcus (1992)}. An option to display the railroad network by railroad will be available {Avison and Wood-Harper (1990), Laurel (1990)}. The colors of the individual railroads will be green for the Burlington Northern, red for the Canadian Pacific, green for the dashed line of the Red River Valley and Western, and red for the dashed line of the Dakota, Missouri Valley and Western. These colors and line types are taken from the 1994 North Dakota Railroad Map used by various state

agencies {Laurel, Marcus}. When a valid railroad branchline is selected, that portion is highlighted and bold {Marcus}.

4. Highway Network: This object will use yellow to display the highway network and will use reasonably heavy lines {Marcus (1992)}. When a valid railroad branchline is selected, the highway impact segment portion is highlighted and bold {Marcus}. After a highway impact analysis, the affected highways are highlighted and color coded to display the highway impact results {Heckel (1991), Laurel (1990)}. The color code will use a spectral order from a bold red through orange to an orange yellow to scale the impacts visually {Marcus}. In addition, the affected highway segments will be widened according to the magnitude of the impact to the highway {Heckel, Marcus}.

5. Grain Elevators: The grain elevators will be displayed on the railroad network and will use a white dot at the elevator location. The dot will be larger than the width of the railroad network lines to be distinguishable, but will use the same color to show the connection of the elevator to railroad network {Heckel (1991), Marcus (1992)}. An option will exist to add place names to the elevators in 10 point, white font {Marcus}.

6. Fertilizer Centers: The fertilizer centers will be displayed on the railroad network and will use a white square at the railroad location. The square will be larger than the width of the railroad network lines to be distinguishable, but will use the same color to show the connection of the fertilizer center to

railroad network {Heckel (1991), Marcus (1992)}. An option will exist to add place names to the fertilizer centers in 10 point, white font {Marcus}.

7. Pop-up Menu: The pop-up menu will have the railroad branchline process colors and will be centered on the screen {Marcus (1992)}. The currently selected menu item will be highlighted and in 12 point, white font {Marcus}.

8. Dialogue box: The dialogue box will have the railroad branchline process colors and will be centered on the screen {Marcus (1992)}. If the analysis terminated message is displayed, the message and a termination symbol will be in red {Marcus}. If the invalid data message is displayed, it will be in red {Marcus}. The analysis proceeds message will be green, and the edit data message will be black {Marcus}.

9. Information box: The information box will have the railroad branchline process colors and will be centered on the screen {Marcus (1992)}. The track progress message will be black with a blue moving sliding gauge to display the percentage of analysis completed {Laurel (1990), Heckel (1991)}.

10. Graphs: The graph object will be incorporated from the main graphic program utilized by the users {Laurel (1990), computer use survey}. Colors in the graphs will use a consistent scale and use the same color across various graphs measuring the same data category {Laurel, Marcus (1992)}. This object will be centered on the HIA screen window and have 2/3 of the

HIA screen area {Marcus}. The icon image and maroon color will be used as a process identifier.

11. Reports: The reports will be in WordPerfect format {Laurel (1990), Marcus (1992), use case surveys}. They will be displayed on the screen as WordPerfect documents.

4.5 Conclusion

This chapter presented user centered design principles and design information to make the IMS prototype a more effective analysis and design tool. By taking advantage of these user centered guidelines, the IMS prototype will present a more visually appealing and constructive user interface. The IMS prototype is an important part of system development, and the effort to make it as productive as possible at the earliest opportunity in the process will lead to more effective system design.

CHAPTER 5. IMS PRELIMINARY SYSTEM IMPLEMENTATION ANALYSIS

5.1 Introduction

This chapter presents the results of a preliminary implementation analysis and survey of potential software development systems and high level IMS functionality done as part of an interim IMS report to the NDDOT. This analysis was done to provide the NDDOT with preliminary background information with which to assess the costs and alternatives available for developing the IMS. Three major categories for development were considered: 1) Database Management Systems (DBMS), 2) Object-Oriented Database Management Systems (OODBMS), and 3) Geographical Information Systems (GIS). Several application development environments were evaluated across these categories. All but one of those application development environment systems used the more structured, traditional type of programming and design. An additional preliminary analysis of an system based on object-oriented technology, but not confined to the three categories, was developed.

The OODBMS category and the object-oriented application development alternative are presented first, followed by the evaluation of four traditional structured programming application development alternatives, two DBMS systems, and two GIS systems. The last alternative outlined is the custom-built system based on object-oriented technology.

These alternatives are presented within the context of a user centered IMS as an information management system, designed to provide a system supporting the administration and to evaluate the state's intermodal transportation system. The alternatives presented here vary in sophistication and their capacity to support intermodal transportation

planning. The specific alternatives are 1) a Delphi-based Object-oriented Database Management System, 2) a dBASE Database Management System, 3) a Visual Basic Database Management System, 4) a Level I ARC/INFO application, 5) a Level II ARC/INFO application, and 6) an object-oriented custom-built system.

5.2 Object-oriented Database Management System

Object-oriented database management systems integrate database technology with object-oriented programming technology. The one presented here is Delphi.

5.2.1 Delphi : An Object-oriented Database Management System

Delphi is an object-oriented programming environment based upon the Object Pascal language. It provides database tools for building database applications, design tools, and an application development environment for Microsoft Windows.

5.2.2 System Overview

A Delphi-based object-oriented database management system would provide the functionality typical of database management systems. In a Delphi-based program, however, the underlying system structure would be represented as objects. The data could be encapsulated into objects containing the database table data and the program code to perform the calculations and manipulations of the data. These objects would be available at runtime for the program to create new objects and abstract data types to respond to state changes in the program as the program models a user centered process.

As a MicroSoft Windows-based program, Delphi provides for the development of a graphical user interface (GUI) having available the resources of the MicroSoft Windows Application Program Interface (API). Delphi also utilizes Object Linking and Embedding

(OLE) technology which provides an opportunity to use the capabilities of other computer programs, such as spreadsheets or word processors. OLE would provide the user access to existing data outside the scope of the IMS and would be a useful enhancement to IMS functionality.

This system would use the XBASE databases. Desktop databases such as Paradox, FoxPro, dBASE, and others may be used. Delphi also supports a client/server edition that works with remote database servers such as Oracle, Sybase, and others. Open Database Connectivity (ODBC) sources may be used for both the desktop and client/server editions. Delphi would be used to develop an interface to these databases.

5.2.3 System Description

The system would be designed to include the following functionality:

- Summary statistics for each database. This includes the standard database statistics and any user-defined statistics. The system would use views and queries plus the additional capability of an enhanced screen display using the Delphi environment.
- Report generation. Custom-designed reports could be developed based upon existing database tables and queries. These reports could be displayed on the screen, faxed or electronically sent to another user, or sent to a local printer.
- Query system development. The Delphi program's query system would require the development of a menu-driven interface to perform query operations and would require significant resources to develop. Alternatively, an SQL interface to a database system like Oracle could be designed.

- Graphs. A set of pre-defined graphs could be developed providing the analyst with an effective visual display of information. Third-party graphing products could be used or accessed allowing more flexibility in designing an effective graphing program component.
- Database maintenance. The database may be maintained at the user site. Common operations, such as adding, editing, or deleting data, would be performed as menu selections or at the custom-designed screen level. The major issue of keeping all instances of the database current must be considered and addressed if this option is developed. In an object-oriented programming system, the capacity to create new database table objects could be utilized during database maintenance.

5.2.4 Database System Structure

The system data structure would include the following databases:

Airports.dbf	Airport Information
95MCDir.dbf	Motor Carrier Directory
Elevat.dbf	Elevator Grain Movement Information
NDLTL.dbf	LTL Trucking Directory
Coal.dbf	Coal Mine Database
Fertiliz.dbf	Fertilizer Database
Transit.dbf	Transit Information

5.2.5 Modeling

Modeling development would be based upon the object orientation of the programming system. Specific objects and new data types (data abstraction) could be created taking advantage of object technology's ability to more closely model the real world system. The resource costs may be higher initially than other systems if the developers need to learn the object-oriented paradigm. However, developing program objects containing modeling capabilities would be less restrictive conceptually than in other systems evaluated here. Sophisticated modeling could be developed, but at a higher cost than the ARC/INFO systems.

5.2.6 GIS

Very limited GIS functionality would be available unless extensive use of an additional language, for instance C++, is incorporated at a lower level in the system.

5.2.7 User Interface

The user interface would be an event-driven graphical user interface. Customized screens would be developed for displaying data from the database and the various performance measures. The Delphi developmental environment lends itself to effective GUI design and rapid GUI prototyping.

5.2.8 System Maintenance and Evolution

A process to maintain the underlying system software and database would need to be established. In addition, any evolution of the system, particularly the database, should be done with the objective of eventual incorporation into a more sophisticated GIS and modeling system.

5.2.9 Summary

A Delphi or another object-oriented programming system should be considered if the IMS becomes complex with the potential for significant evolution. An object-oriented software system could be designed to reduce the complexity with the use of independent objects. These independent objects can be used to manage program complexity and can be reused and modified to create new objects in the system as the system evolves. However, for a large IMS, an object-oriented development system offering more flexibility and extensibility should be considered. It is important to note that this system would not offer as much decision support as other alternatives.

5.3 DataBase Management Systems: Level I

A database management system provides the functionality to support and use a relational database. Two levels of possible development are presented here.

5.3.1 dBASE Database Management System

dBASE is a program long used by the NDDOT and is evaluted here in consideration of the advantage the extensive knowledge of dBASE would provide.

5.3.2 System Overview

A dBASE database management system would be a standard relational database information system. It would incorporate the intermodal databases into a dBASE application and provide users with conventional database access to the intermodal database. The system would support data queries through the dBASE query system and would use customized dBASE data display screens. DOS and MicroSoft Windows versions could be developed as well as a networked version.

The system would provide views of the intermodal inventory and any predefined intermodal performance measures. The intermodal analyst would be able to use this system to access basic, static data. It is being presented as a typical database management system developed in an environment familiar to many NDDOT users. This takes advantage of existing NDDOT skills and allows these users to utilize the database in their own dBASE environment.

5.3.3 System Description

The system would be designed to include the following functionality:

- Standard dBASE functionality. The system would allow easy access to the existing, underlying dBASE functionality. A skilled dBASE user could use this function to go directly to the dBASE environment.
- Summary statistics for each database. Predefined statistics could be developed for some or all of the databases. These statistics would include summation, averages, weighted averages, and others. This would provide the analyst with quick access to a set of system defined statistics requiring the development of database views and queries.
- Report generation. Custom-designed reports could be developed based upon existing database tables and queries. These reports could be displayed on the screen or sent to a local printer.
- Query system development. The program's query system could adopt one or more of three different approaches. The first approach would drop the user into the dBASE environment for performing queries. This approach uses

fewer developmental resources, but requires familiarity with using dBASE. A second approach would be to develop an interface to the dBASE query system that reduces, but does not eliminate, the need for dBASE familiarity. The third approach is the development of a new, menu-driven interface to perform query operations.

- Graphs. A set of system-defined graphs could be developed, particularly for a MicroSoft Windows version of the software. This would provide the analyst with a more visual source of information.
- Database maintenance. The database may be maintained at the user site. Common operations, such as adding, editing, or deleting data, could be performed in two ways. The first, requiring dBASE familiarity, would drop the user into the dBASE environment for these operations. The second, requiring more resources to develop, would perform these operations at the custom-designed screen level. The major issue of keeping all instances of the database current must be considered and addressed if this option is developed.

5.3.4 Database System Structure

The database structure would be designed to allow access to individual databases (those outlined in Section 2.2.3) and to support utilization of all databases as a whole. The structure would provide the framework for the integration of the databases to facilitate queries and reports based upon one or more of the databases.

The design of the database system would especially consider the geographic reference points common to the databases and transportation analysis. An example of this would be the development of relationships among the databases based upon a geographic region (county) or location (latitute, longitude) that expedites the query system.

5.3.5 Modeling

Modeling would have limitions in this system. Considering developmental resources, it may be limited to sensitivity analysis of selected performance measures. This would be accomplished by manipulating the data elements serving as parameters into the performance measure analysis. Additional models would require the development of separate program modules accessing the database and perfoming calculations and analysis. Sophisticated modeling, such as routing algorithms, facility location and other analytical procedures, do not readily lend themselves to this system when compared to other alternatives. The results of these models would be represented in graphs or reports.

5.3.6 GIS

There would be no GIS functionality.

5.3.7 User Interface

The user interface would consist of a graphical menu structure accessing the system's functionality. The MicroSoft Windows version would incorporate a more truly graphical user interface. Customized screens would be created for displaying data from the database and the various performance measures.

5.3.8 System Maintenance and Evolution

A process to maintain the underlying system software and database would need to be established. In addition, any evolution of the system, particularly the database, would be done with the objective of eventual incorporation into a more sophisticated system

5.3.9 Summary

The dBASE management system would be the most elementary system presented and also the least expensive to develop. This system would be a bare bones approach providing a rudimentary information system managed from a database perspective. It would essentially be an information tool, not an analysis tool, in support of transportation planners and decision makers. As such, it would not be as effective a decision support system as other alternatives.

5.4 DataBase Management Systems: Level II

Level II of the database management systems provides a more sophisticated system.

5.4.1 Visual Basic Database Management System

Visual Basic is an application development system providing rapid prototyping and strong connectivity to MicroSoft Windows. Numerous third-party software solution providers have extended Visual Basic by developing many add-on components. Functionality not inherent to Visual Basic is readily available, facilitating development in this environment.

5.4.2 System Overview

A MicroSoft Windows-based Visual Basic database management system would have many of the characteristics of the dBASE system. However, much more is possible in Visual

Basic. A major difference between the two lies in the development of the user interface or graphical user interface (GUI). In a Visual Basic system, or a similar developmental environment such as Powerbuilder, the GUI has more potential for development. Visual Basic and related add-on developmental tools provide a richer set of possibilities to meet user expectations for the user interface. Visual Basic is more closely coupled with Object Linking and Embedding (direct active program access to other programs such as MicroSoft's Word or Excel) when compared to dBASE for MicroSoft Windows. Object Linking and Embedding (OLE) provides an opportunity to use the capabilities of other computer programs and may be useful for accessing spreadsheets or word processors.

This system would use the dBASE databases. Visual Basic would be used to develop an interface to these databases.

5.4.3 System Description

The system would be designed to include the following functionality:

- Summary statistics for each database. This system would have the same functionality as described in Section 5.3.1 plus the additional capability of an enhanced screen display using the Visual Basic environment.
- Report generation. This system would have the same functionality as described in Section 5.3.1.
- Query system development. The Visual Basic program's query system would require the development of a new, menu-driven interface to perform query operations. This would use significant resources to develop. Alternatively, an SQL interface to a database system like Oracle could be designed.

However, a third-party product could be investigated to provide this functionality at a reasonable cost.

- Graphs. This system would have the same functionality as described in Section 5.3.1. However, more third-party graphing products could be used or accessed allowing more flexibility in designing an effective graphing program component.
- Database maintenance. The database may be maintained at the user site. Common operations, such as adding, editing, or deleting data, would be performed as a menu selection or at the custom designed screen level. The major issue of keeping all instances of the database current must be considered and addressed if this option is developed.

5.4.4 Database System Structure

The database structure would be designed as described in Section 5.3.1.

5.4.5 Modeling

Modeling would be limited in this system, but not as limited as in the dBASE system. Developing program modules containing modeling capabilities would be less restrictive as in the dBASE system, and representing the results of a model would be enhanced because of a more flexible GUI environment. Again, as in the dBASE system, sophisticated modeling could be developed, but at a higher cost than other alternatives.

5.4.6 GIS

This system would have very limited GIS functionality at a high resource cost.

5.4.7 User Interface

The user interface would be an event-driven graphical user interface. Customized screens would be developed for displaying data from the database and the various performance measures. The Visual Basic developmental environment lends itself to effective GUI design and rapid GUI prototyping, streamlining the developmental process.

5.4.8 System Maintenance and Evolution

A process to maintain the underlying system software and database would need to be established. In addition, any evolution of the system, particularily the database, can be done more easily than the dBASE system, particularly with the objective of eventual incorporation into a more sophisticated GIS and modeling system.

5.4.9 Summary

The Visual Basic management system would be similar to the dBASE system. It is presented here to consider the possibility of more extensive user interface development and some limited potential for better modeling support. This system would not include the underlying, built-in functionality of an application developed in dBASE thereby increasing some development costs. However, when compared to the dBASE system, it offers some areas of improvement while making development more difficult in others. In considering this alternative, the importance of the user interface will be a primary concern. Overall, this system would not offer as much of a decision support system as other alternatives.

5.5 GIS-based Systems

The North Dakota Department of Transportation is supporting the development of an ARC/INFO Geographical Information System (GIS). This system could be used to develop an IMS. Two levels of system development are presented here.

5.5.1 Level I ARC/INFO System

The NDDOT has installed and is continuing development of an ARC/INFO GIS system. This Level I evaluation is presented to take advantage of this system.

5.5.1.1 System Overview

An ARC/INFO IMS system would be an integrated geographic information system. As presented here, a Level I system would provide all the GIS capabilities common to ARC/INFO applications and systems. In the Level I system, the transportation analyst would have an underlying relational database management system containing geographic attribute information linked to a sophisticated mapping display. This combination provides an excellent visual information source for evaluting and analyzing transportation networks and transportation facility location. The Level I system would also take advantage of ARC/INFO's built-in modeling capabilities and provide intermodal transportation modeling applications written in ARC Macro Language (AML).

5.5.1.2 System Description

The system would be designed to incorporate the following ARC/INFO GIS mapping and spatial analysis capabilities:

- Link-node topology for representing/displaying transportation networks.
- Route and network segment calibration and delineation.

- Multiple facility location displays.
- Multiple GIS layering representing segments of the transportation network.
- Least-cost path determinations.
- Transportation resource allocation.
- Network modeling.
- Vehicle routing analysis.
- Spatial relationships representation.
- Geographic and database attribute display.

The system also would include the following:

- Summary statistics for each database.
- Enhanced report generation.
- Geographic query system development.
- Graphs.
- Database maintenance. The database may be maintained at the user site. The major issue of keeping all instances of the database current must be considered and addressed if this option is developed.

5.5.1.3 Database System Structure

The database structure would be designed to integrate into the ARC/INFO GIS and spatial analysis capabilities. The database would be formatted and structured as a spatial database. Data conversion from the dBASE databases would be an important part of the process of defining this database system.

5.5.1.4 Modeling

Modeling would be extensive in this system. In particular, the spatial analysis capabilities provided by the software environment would be utilized to develop specific intermodal transportation applications. ARC/INFO clearly provides the most capable modeling environment considered by this analysis.

5.5.1.5 GIS

Extensive GIS functionality would be available in this system. Multiple layers of the intermodal transportation network would be developed providing the analyst with sophisticated visual displays. This would give the analyst a set of comprehensive visual tools to evaluate, examine, and utilize the intermodal network. ARC/INFO GIS capabilities provide an opportunity for using visual information unmatched by the other systems analyzed in Chapter 5.

5.5.1.6 User Interface

The user interface would consist of three different levels. A command-level interface is available, and AML can be used to create a custom-designed graphic user interface. A third level is the built-in ARC TOOLS, a GUI interface to the system. In addition to the possibility of third party interface applications, there is a rich variety of user interface selections.

5.5.1.7 System Maintenance and Evolution

System maintenance and evolution would require an ARC/INFO specialist. The specialized nature of the system would require a more extensive maintenance process, but

should be considered in the context of the much richer functionality of an ARC/INFO system.

5.5.1.8 Summary

This system gives the intermodal analyst a variety of analytical tools inside a sophisticated visual display system. ARC/INFO does provide an extensive environment for the development of a decision support system. However, the development of this type of decision support system will have a higher resource cost than the alternatives. The inclusion of GIS and modeling capabilities in an IMS do provide many analytical possibilities and should be strongly considered.

5.5.2 Level II ARC/INFO System

A Level II ARC/INFO system would include all the capabilities of the Level I ARC/INFO system. A Level II system would extend the modeling capabilities of the IMS. Various analytical models, now under consideration, would be developed to give the user additional analytical support. These models, drawing upon some of the unique characteristics of the underlying intermodal database, would be custom designed and focused on providing a decision support environment more specific for an IMS in North Dakota. While involving more resources to develop, it provides more for the transportation analyst and decision maker.

5.6 Object-oriented Software System

Object-oriented programming utilizes a different approach to the development of software and is discussed to provide a background for describing some potential advantages in using this type of programming environment. In the traditional, procedure-oriented

programming approach, a software system provides an algorithm or a series of specific steps to be performed. In the object-oriented approach, the program describes a system of fully functional independent entities called objects which are allowed to interact with each other. Object-oriented approaches can make large programs simpler, easier to change, easier to maintain, extensible, and reusable.

The objects in an object-oriented system are the basic units of the system and encapsulate both data and operations for manipulating that data within one program unit. An object's internal structure may include new abstract data types specific to the system the program is modeling and mechanisms for creating new objects with identical or similar characteristics. This object and programming structure gives the software the potential to model a system more closely than the traditional approach and also provides a mechanism for the evolution of the software to create and modify new objects in the system.

The advantages of using an object-oriented programming approach include code reuse and a programming model that may be used to reduce the complexity in a system. Code reuse in object-oriented programming is provided by the ability to reuse user-defined abstract data types and the capability to create new objects from existing objects. The object programming model provides a way to design a software system that parallels the objects in a real world system and can reduce complexity by the inclusion into the system of independent objects. These independent objects provide the developer the capability of isolating working code and, through inheritance, creating new independent objects with similar characteristics.

This programming approach should be considered if the IMS system evolves into a large, complex system that models a system undergoing continuous evolution and if the software system will require a lot of maintenance. The object-oriented system presented here will be based upon an object-oriented programming language such as C++. Other object-oriented programming systems, such as Powerbuilder, are available. If the IMS system will be developed accessing GIS functionality, a programming language such as C++ should be considered for object-oriented system development and modeling. C++ would be used for low level programming providing or connecting to GIS functionality.

CHAPTER 6. CONCLUSION

This paper sought to accomplish three objectives, each to be used in the development of a transportation analysis software system. The first objective was to select a user centered object-oriented analysis and design methodology for systems analysis of the software. To accomplish this objective, a survey and an examination of object-oriented analysis and design methodologies were done. The first step in the survey process was the development of the basic principles and concepts of object-oriented technology. This step included an introduction to object-oriented technology, a discussion of the major object-oriented concepts, and an overview of the historical evolution of the technology. Second, the evaluation of several, major object-oriented analysis and design methodologies was completed. This evaluation considered the main analysis and design approaches and techniques of each methodology within the context of user centered software design. Third, a user centered object-oriented analysis and design methodology was selected, Jacobson's use case approach.

Jacobson's use case methodology focuses immediately on the user. This is in contrast to the other methodologies that were evaluated and is done with the development of use cases. Use cases are the analytic technique of identifying what and how the user will use the system. A use case is developed by first identifying the actors of a system and then determining their uses of the system.

This approach is user centered because it initiates the analysis with the user from the user's perspective and remains centered on the user as the use case drives all the analysis and design modeling. All the analysis and modeling validation is done against the use case

model. The entire software system and systems analysis is developed from the user's perspective.

Others, including leading object-oriented methodologists James Rumbaugh (1994) and Grady Booch (1994), have also recognized this approach's value as a user centered methodology. Rumbaugh and Booch are incorporating the use case into their upcoming Unified Method and recommend its use as a user centered approach to object-oriented analysis and design.

A second objective of the study was to initiate the application of the use case approach to the Intermodal Management System (IMS), the transportation analysis system being considered for development. The application of the use case approach was done at two system levels. A high level analysis was used to identify the actors in the system and six high level use cases. A lower level system analysis included a more detailed use case example which was developed to provide an example of an analysis of greater depth.

The first step in initiating the application of the use case approach to the IMS was the identification of the system actors. The study augmented and expanded upon Jacobson by introducing a priority ranking of the actors. The identification of the IMS use case actors revealed that one primary actor, the transportation planner, would dominate system utilization. The priority ranking of the actors illuminated this and allowed for use case development to concentrate on the transportation planner.

The second step in the application process was the identification and the development of the use cases. This was done with interviews and background surveys of the primary actor as well as the utilization of an interim IMS report to the North Dakota

Department of Transportation. Six high level use cases were developed. The American Management System stratification of use cases discussed in Chapter 2 was adopted to expand and detail the use cases while remaining strongly focused on the user. This stratification was utilized to provide greater depth to an example use case connecting the use case more closely to object analysis and design.

The third step in the application process was the development of object class specifications for the expanded use case. Twelve object classes were developed and were used to demonstrate the inheritance, encapsulation, and reuse of IMS objects. These objects were also used in a comparison of an object-oriented approach to a data-driven approach. The comparison was done to highlight the possible benefits the application of user centered object-oriented technology offered and considered the issues of understanding the user and managing system complexity. The possible benefits include a greater understanding of the user by development from the user's perspective, the management of system complexity through the use of abstraction, and the extensibility and reusability of objects.

The final step in the application process was making user centered design decisions for an IMS prototype. The prototype process is used by many design methodologies to evaluate early analysis and design decisions. This study extended and enhanced the prototype process by developing a user centered design framework for the IMS prototype. This framework included examining several user centered design approaches, including those that address the issue of interface design and human-computer interaction, as well as developing a computer use survey of the likely IMS users. These user centered design principles were applied by making IMS prototype design decisions for the example use case.

These user centered design decisions will result in a more effective prototype, thereby enhancing the application process.

The last objective of the study was to survey alternative software development systems for possible IMS implementation. To accomplish this objective, the North Dakota Department of Transportation was provided with an assessment of alternatives for IMS software development. A series of six potential software systems were examined and presented for NDDOT's consideration when evaluating the factors affecting IMS implementation.

The three major categories evaluated for possible IMS software development were 1) Database Management Systems (DBMS), 2) Object-Oriented Database Management Systems (OODBMS), and 3) Geographical Information Systems (GIS). Several application development environments were examined across these categories. An additional preliminary analysis of a system based on object-oriented technology, but not confined to the three categories, was developed.

The OODBMS category and the object-oriented application development alternative were presented first, followed by the evaluation of four traditional structured programming application development alternatives, two DBMS systems, and two GIS systems. The last alternative outlined was the custom-built system based on object-oriented technology.

These alternatives were presented in response to the North Dakota Department of Transportation, for its consideration of the alternatives possible for IMS implementation.

As such, they formed part of the user centered design process by responding to the user's perspectives and requirements.

The author also formed several conclusions concerning the applicability of the paper's approach to the development of an Intermodal Transportation System. First, the utilization of a user centered approach adopted by the study is recommended. The transportation professional, working inside an intermodal context, will be presented with a variety of transportation analysis scenarios and tasks. The adoption of a user centered design approach will result in a software system concentrated on the presentation and completion of these tasks rather than on the complexity of determining how the data are manipulated by the system to perform a myriad of tasks. This focus will minimize the effort required by the transportation professional to use the system.

Second, the author recommends that object-oriented technology be used only for large, complex installations of an IMS. The benefits derived from object-oriented technology are most noticeably gained when the complexity of a system is a major issue. The IMS, with the many different databases and analysis scenarios, has the potential to be a complex system; and adopting object-oriented technology for large IMS installations offers substantial benefits. However, a small IMS installation would not benefit from object-oriented technology.

Third, the author concludes that departments of transportation can benefit from the development of an object-oriented IMS. Object-oriented technology provides an effective vehicle for software evolution and, as a IMS matures, will meet the need for transportation professionals to adopt emerging analytical standards and practices. An object-oriented IMS,

with a user centered design, will provide an efficient software system adapted to meet the changing requirements of intermodal transportation analysis.

BIBLIOGRAPHY

Abbott, R. J. 1983. Program Design by Informal English Descriptions. Communications of the ACM, Vol. 26, No. 11, pp. 882 - 894.

Armour, F., Boyd, L., and Sood, M. 1995. Use Modeling Concepts for Large Business System Development. American Management Systems, OOPSLA Workshop on Use Cases, Preliminary Program, October 15, 1995, Austin, Texas, http://www.unantes.univ-nantes.fr/usecase/rootExtension.html.

Avison, D. and Wood-Harper, A. 1990. MULTIVIEW: An Exploration in Information Systems Development. The Alden Press, Oxford, England.

Baecker, R. and Marcus, A. 1990. Human Factors and Typography for More Readable Programs. Addison-Wesley Publishing Co., Reading, Massachusetts.

Barton-Aschman Associates, Inc. 1993. Study Design for the New Mexico Statewide Intermodal Transportation Plan. New Mexico State Highway and Transportation Department, Santa Fe, New Mexico, June 9, 1993.

Berard, E. 1993. Essays on Object-Oriented Software Engineering, Volume 1. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Berard, E. 1995. Be Careful With "Use Cases." The Object Agency, Inc., Gaithersburg, Maryland, http://www.toa.com/pub/use case.txt.

Booch, G. 1982. Object Oriented Design. Ada Letters, Vol. I, No. 3, pp. 64-76.

Booch, G. 1986. Object Oriented Development. IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, February 1986, pp. 211 - 221.

Booch, G. 1991. Object-Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company, Inc., Redwood City, California.

Booch, G. 1994. Object-Oriented Analysis and Design with Applications. Second Edition. The Benjamin/Cummings Publishing Company, Inc., Redwood City, California.

Booch, G. and Rumbaugh, J. 1995. Unified Method for Object-Oriented Development, Documentation Set, Version 0.8. Rational Software Corporation, Santa Clara, California.

Borland International Inc. 1995. Delphi for Windows. Borland International, Scotts Valley, California.

Borland International Inc. 1995. dBASE for Windows. Borland International, Scotts Valley, California.

Brinkkemper, S., Hong, S., Bulthuis, A., and van den Goor, G. 1995. Object-Oriented Analysis and Design Methods. University of Twente, Enschede, The Netherlands, http://wwwis.cs.utwente.nl/dmrg/OODOC/oodoc/oo.html.

Brown, J. S. 1986. From Cognitive to Social Ergonomics. In User Centered System Design: New Perspectives of Human-Computer Interaction, Norman, D. and Draper, S. Editors. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Card, K., Moran, T., and Newell, A. 1983. The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Carter, D. 1993. Development of an Intermodal Transportation Management System for California. Book, Allen & Hamilton, Los Angeles, California.

Coad, P. and Nicola, J. 1993. Object-Oriented Programming. P T R Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Coad, P. and Yourdon, E. 1991a. Object-Oriented Analysis. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Coad, P. and Yourdon, E. 1991b. Object-Oriented Design. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Dahl, O. and Nygaard, K. 1966. SIMULA -- an ALGOL-Based Simulation Language. Communications of the ACM, Vol. 9, No. 9, pp. 671 - 678.

Davis, G. 1974. Management Information Systems: Conceptual Foundations, Structure and Development. McGraw-Hill, New York.

Entsminger, G. 1990. The Tao of Objects. M&T Books, New York.

Environmental Systems Research Institute, Inc. ARC/INFO - An Integrated Answer for Transportation Planning and Management. Redlands, California.

Graham, I. 1994. Object-Oriented Methods, Second Edition. Addison-Wesley, Wokingham, England.

Hansen, T. and Miller, G. 1995. Requirements Definition and Verification Through Use Case Analysis and Early Prototyping. OOPSLA Workshop on Use Cases, Preliminary Program, October 15, 1995, Austin, Texas, http://www.unantes.univ-nantes.fr/usecase/rootExtension.html.

Heckel, P. 1991. The Elements of Friendly Software Design, The New Edition. SYBEX, Inc., Alameda, California.

HOOD Working Group (1989). HOOD Reference Manual. Issue 3.0 European Space Agency, Noordwijk, The Netherlands.

Jacobson, I. 1992. Object-Oriented Software Engineering, A Use Case Driven Approach. Addison-Wesley Publishing Company, Reading, Massachusetts.

Kay, A. 1977. Microelectronics and the Personal Computer. Scientific American, Vol. 237, No. 3, pp. 230-244.

Kirk, F. 1973. Total System Development for Information Systems. John Wiley & Sons, Inc., New York.

Laurel, B. Editor. 1990. The Art of Human-Computer Interface Design. Addison-Wesley Publishing Company, Reading, Massachusetts.

Marcus, A. 1992. Graphic Design for Electronic Documents and User Interfaces. ACM Press, New York.

MicroSoft Corporation. 1993. MicroSoft Visual Basic, Programming System for Windows, Version 3.0. MicroSoft Corporation, Redmond, Washington.

Norman, D. and Draper, S. Editors. 1986. User Centered System Design: New Perspectives of Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Peterson, G. 1987. Object-Oriented Computing, Volume 1: Concepts. Computer Society Press of the IEEE, Washington, District of Columbia.

Peterson, G. 1987. Object-Oriented Computing, Volume 2: Implementations. Computer Society Press of the IEEE, Washington, District of Columbia.

Pfleeger, S. 1991. Software Engineering, The Production of Quality Software, Second Edition. Macmillan Publishing Company, New York.

Rehe, R. 1974. Typography: How to Make It Most Legible. Design Research International, Carmel, Indiana.

Rosenberg, D. 1995. A Unified Object Modeling Approach -- Integrating Jacobson, Rumbaugh, and Booch Methods. ICONIX Software Engineering, Inc., Santa Monica, California, http://www.iconixsw.com/iconix/Spec Sheets/UnifiedOM.html.

Rumbaugh, J., et al. 1991. Object-Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, New Jersey.

Rumbaugh, J. 1994. Getting Started, Using Use Cases to Capture Requirements. Journal of Object-Oriented Programming, Vol. 7, No. 5, pp. 8 - 23.

Schmidt, N. 1996. Personal communication. North Dakota Department of Transportation, Bismarck, March 1996.

Shneiderman, B. 1992. Designing the User Interface, Strategies for Effective Human-Computer Interaction. Addison-Wesley Publishing Company, Reading, Massachusetts.

Sommerville, I. 1992. Software Engineering, Fourth Edition. Addison-Wesley Publishing Company, Reading, Massachusetts.

Tolliver, D. 1996. Personal communication. Upper Great Plains Transportation Institute (UGPTI). North Dakota State University, Fargo, March 1996.

United States Department of Transportation. 1993. A Summary of the Intermodal Surface Transportation Efficiency Act. Government Publications Office: 1993 0-347-721 QL 3, Washington, District of Columbia.

United States Department of Transportation, Federal Highway Administration. 1993. Intermodal Surface Transportation Efficiency 1991: Selected Fact Sheet. Washington, District of Columbia.

United States Department of Transportation, Federal Highway Administration. 1993. 23 CFR Part 500, et al. Management and Systems; Proposed Rule. Washington, D.C.

Upper Great Plains Transportation Institute (UGPTI). 1995. North Dakota Intermodal Management System Phase II Report. North Dakota State University, Fargo.

Wilkinson, N. 1995. Object Analysis, CRC Cards as Input to a Formal Methodology. Software Development, Vol. 3, No. 11, pp. 45-55.

Yourdon, E. 1979. Classics in Software Engineering. YOURDON Press, New York.

APPENDIX A

IMS BACKGROUND SURVEYS

IMS Actor Identification and Priority Specification

Note: An actor is a user of the IMS or an entity interacting with the IMS.

System Context:HUMAN ACTORS

These people have been identified as users of the IMS. Please indicate whether they are primary, direct users of the system or secondary, occasional or system-support users? Also, please rank them on a scale of 1 to 9 with 1 being the most important, most direct user.

Planning Division	Primary User	Secondary User	Priority
Division Head	☐	☑	6
Transportation Planners	☑	☐	1
Traffic Operations	☐	☐	4
Mapping	☑	☐	2
Rail Program	☑	☐	3
Other Divisions:			
Operations	☐	☑	7
Program&Proj. Dev.	☐	☑	8
Secondary Roads	☐	☑	9
Upper Management	☐	☑	5

System Context:OTHER SYSTEM ACTORS

These systems have been identified as users of the IMS or as systems interacting with the IMS. Please indicate whether they are primary, direct users of the system or secondary, occasional or system-support users? Also, please rank them on a scale of 1 to 4 with 1 being the most important, most direct user.

	Primary User	Secondary User	Priority
Pavement Management System	☐	☑	2
GIS System	☑	☐	1
Transportation Data Systems	☐	☑	3
Traffic Operations Systems	☐	☑	4

IMS Transportation Planner Use Case Specification

Note: A use case, as described by Jacobson (1995), is a sequence of transactions in a system whose task is to yield a measurable value to an individual actor of the system. This document, by recording how transportation planners will use the IMS, what would they want the system to do and what would they get back from the system, will be used to develop use cases for the transportation planner.

The AMS approach of four levels of use cases is followed with the exclusion for the last use case scenario, the abstract case, which is considered elsewhere.

High Level Use Case

A high level use case is one describing broad or major system behaviors. Previous discussion and reports suggest that the IMS may contain these major system activities:

Decision Support: The IMS would be used for direct decision making. The IMS becomes an active part of decision processes and decisions are made based directly on the IMS.

Planning Information: The IMS provides information for planning activities and becomes part of the planning context.

Information Service: The IMS becomes a general repository of information and provides query and general information services.

Reports: IMS reports become a part of work activities and are used for other information-based activities.

These definitions apply to the first question.

What will be your major uses of the IMS?

- ☐ Decision Support
- ☑ Planning Information
- ☑ Information Service
- ☑ Reports
- ☐ Other

What high level functionality will you use in the IMS?

- ☑ Information Query System
- ☑ Data storage and retrieval
- ☑ Performance Measure and Display
- ☑ Report generation
- ☐ Decision Support/Modeling
- ☐ Other (Please list)_____

Expanded Use Cases

An expanded use case elaborates and expands on the high level use cases.

What functionality do you want from the Information Query System?

- ☑ Standardized (use existing Xbase or other database query system) user interface
- ☐ Custom-built query user interface
- ☑ Print query result
- ☑ Store queries
- ☑ Predefined queries
- ☑ Multiple database queries
- ☐ Export query or query result in specialized record format
  Possible formats:
  - ☐ MicroSoft Word
  - ☑ WordPerfect
  - ☑ Xbase
  - ☐ MSAccess
  - ☐ Excel
  - ☐ Other _____
- ☐ Other (Please list)_____

What functionality do you expect from the Performance Measure Calculation and Display?

- ☑ Custom performance measure interface
- ☑ Graphical screen display
- ☑ Generate report
- ☐ Export result in specialized record format
  Possible formats:
  - ☐ MicroSoft Word
  - ☑ WordPerfect
  - ☑ Xbase
  - ☐ MSAccess
  - ☐ Excel
  - ☐ Other _____
- ☐ Other (Please list)_____

What functionality do you want for Data Storage and Retrieval?

- ☑ Multi year storage and retrieval
- ☑ Long term storage
- ☑ Security and encryption
- ☑ Store and retrieve data in specialized record formats
  Possible formats:

- ☐ MicroSoft Word
- ☑ WordPerfect
- ☑ Xbase
- ☐ MSAccess
- ☐ Excel
- ☐ Other

☑ Storage media

Possible media:

- ☐ Local Hard drive
- ☑ Networked hard drive
- ☐ CD-ROM
- ☐ Tape
- ☐ Excel
- ☐ Other_____

☐ Other (Please list) _____

What functionality do you want for Report Generation?

☑ Standard reports
☑ Custom-built report functionality
☑ Print reports
☑ Store reports/report database
☑ User editing of reports
☑ Report format

Possible formats:

- ☐ MicroSoft Word
- ☑ WordPerfect
- ☑ Xbase
- ☐ MSAccess
- ☐ Excel
- ☐ Other_____

☐ Other (Please list) _____

What other functionality must be included in the IMS?

☑ Electronic Data Interchange
☑ Fax/modem services
☐ Other (Please list)_____

Detailed Use Cases: Performance Measure Calculation

What databases will you use in the IMS?

- ☑ Coal Handling Facilities
- ☑ Airports
- ☑ Transit Facilities
- ☑ Fertilizer Distribution Centers
- ☑ Grain Elevators
- ☑ Major Motor Carrier Terminals
- ☑ Sugar Beet Collection and Processing Facilities
- ☐ Other (Please list)_____

What performance measures should be included in the IMS?

Coal Handling Facility Database:

- ☑ Basic traffic characteristics
- ☐ Other (Please list)

_____

_____

Airport Database:

- ☑ Basic traffic characteristics
- ☐ Freight/Operations Efficiency Measure
- ☐ State Highway Location Metric
- ☑ Scheduled air service
- ☐ Other (Please list)

_____

_____

Grain Elevator Database:

- ☑ Basic traffic characteristics
- ☑ Modal access and share
- ☑ Rail shipment characteristics
- ☑ Truck shipment characteristics
- ☑ Elevator shipping capacity and annual volume
- ☐ Other (Please list)

_____

_____

Fertilizer Distribution Database:

- ☑ Basic traffic characteristics
- ☐ Car handling capacity
- ☑ Storage capacity
- ☐ Other (Please list)

_____
_____

Motor Carrier Database:

- ☑ Basic traffic characteristics
- ☑ Truck Type Unloading Time Metric
- ☐ Other (Please list)

_____
_____

Transit Facility Database:

- ☑ Basic traffic characteristics
- ☑ Operational characteristics
- ☐ Other (Please list)

_____
_____

Sugar Beet Collection and Processing Database:

- ☑ Basic traffic characteristics
- ☑ Other (Please list)
  Storage capacity
  _____
  _____

Systemwide:

- ☑ Commodity shipment routing characteristics, highway, rail and air
- ☑ Highway structural demands by freight shippers
- ☑ Support Pavement Management System (PMS)/Traffic Management System (TMS)
  - ☑ Compute intermodal average daily truck trips
  - ☑ Identify commodities originated/terminated on specific highway sections
  - ☑ Estimate intermodal facility generated truck traffic by highway section
  - ☑ Project traffic demand with modal shift
  - ☐ Other (Please list). _____

Systemwide:

- ☐ Computer intermodal facility-specific ESALS
- ☐ Facility throughput
- ☐ Facility modal access
- ☑ Facility/Modal transfer time
- ☑ Service Frequency
- ☑ Waiting Time
- ☑ FRA Track Classification for rail access

What databases will be updated?

- ☑ Coal Handling Facilities
- ☑ Airports
- ☑ Transit Facilities
- ☑ Fertilizer Distribution Centers
- ☑ Grain Elevators
- ☑ Major Motor Carrier Terminals
- ☑ Sugar Beet Collection and Processing Facilities

System High Level Use Case

When will the IMS databases be updated?

- ☐ Daily
- ☐ Occasionally
- ☐ Annually
- ☐ Never

NDDOT Comment: Every 3-5 years.

How will these databases be updated?

- ☐ User input
- ☐ Outside file creation and installation
- ☑ EDI Updated files captured by EDI process.
- ☐ Interface to other software

Please list: _____

Who will update the databases?

- ☐ User
- ☐ System Administrator
- ☑ Outside consultant

System Context and Requirements Interview

System Context:HUMAN

Which groups of people will include the IMS as a part of their work activity or as a contributor to their work product?

		Comments
Planning Division		
Division Head	☑	
Transportation Planners	☑	
Transportation Data	☐	
Traffic Operations	☑	Track counts, changes, intermodal mode shifts.
Urban Program	☐	
Mapping	☑	Through GIS, also incorporate inventory data.
Rail Program	☑	Use for updating rail program, background data.
Transit/Special Studies	☐	Depend more on TMS.
Other Divisions:		
Operations	☑	Districts interested in specific projects.
Design	☐	
Bridge	☐	
Materials&Research	☐	
Program&Proj. Dev.	☑	Prioritizing of district projects.
Right of Way	☐	
Secondary Roads	☑	Limited, impact of projects on county roads.
Upper Management	☑	Category added by NDDOT. Use for what if questions, where are traffic generators.

How will the IMS contribute to the work of these groups of people? The categories are explained below.

Planning Division	Decision Support	Planning Information	Information Service	Reports
Division Head	☐	☑	☑	☐
Transportation Planners	☐	☑	☑	☑
Transportation Data	☐	☐	☐	☐
Traffic Operations	☐	☑	☐	☐
Urban Program	☐	☐	☐	☐
Mapping	☐	☑	☐	☐
Rail Program	☐	☑	☐	☐
Transit/Special Studies	☐	☐	☐	☐
Other Divisions:				
Operations	☐	☑	☐	☐
Design	☐	☐	☐	☐
Bridge	☐	☐	☐	☐

Materials&Research	☐	☐	☐	☐
Program&Proj. Dev.	☐	☑	☐	☐
Right of Way	☐	☐	☐	☐
Secondary Roads	☐	☐	☐	☐
Upper Management	☑	☐	☐	☑

Category added by NDDOT. Use for special projects coming from upper management. IMS will not be used directly by upper management but used by those developing information for upper management decisions. The IMS will support information development across the other three categories.

Decision Support: The IMS would be used for direct decision making. The IMS becomes an active part of decision processes and decisions are made based directly on the IMS.

Planning Information: The IMS provides information for planning activities and becomes part of the planning context.

Information Service: The IMS becomes a general repository of information and provides query and general information services.

Reports: IMS-generated reports become a part of work activities and are used for other information-based activities.

What is the general computer skill level of these users? ☐ Poor ☑ Average ☐ Good

System Context:COMPUTER SYSTEMS

Will the IMS be a stand-alone, networked or client-server system?

- ☐ Stand-alone
- ☑ Networked — Data items become part of GIS.
- ☐ Client-server system

With what computer systems will the IMS directly interact or communicate?

- ☑ Pavement Management System (PMS)
- ☑ GIS System
- ☐ Bridge Management System
- ☑ Transportation Data Systems
- ☑ Traffic Operations Systems
- ☐ Mapping Systems
- ☐ None
- ☐ Other (Please list)

_____

_____

What computer systems will the IMS provide or receive input/output electronic data?

- ☑ Pavement Management System (PMS)
- ☑ GIS System

- ❑ Bridge Management System
- ❑ Transportation Data Systems
- ❑ Traffic Operations Systems
- ❑ Mapping Systems
- ❑ None
- ❑ Other (Please list)

_____

_____

What general computer record formats must the IMS read or write?

- ❑ MSAccess
- ☑ dBASE
- ❑ FoxPro
- ❑ Lotus 1-2-3
- ❑ Excel
- ❑ Quattro Pro
- ❑ MicroSoft Word
- ☑ WordPerfect
- ❑ ASCII
- ❑ None
- ❑ Other (Please list)

_____

_____

System Requirements

What general functionality will the IMS provide?

- ☑ Information Query System
- ☑ Data storage and retrieval
- ❑ Performance Measure and Display
- ☑ Report generation
- ☑ Decision Support/Modeling
- ❑ Other (Please list)

_____

_____

What functionality is necessary for an Information Query System?

- ☑ Standardized (use existing Xbase or other database query system) user interface
- ❑ Custom-built query user interface
- ☑ Print query result
- ☑ Store queries
- ☑ Predefined queries

- ☑ Multiple database queries
- ☐ Export query or query result in specialized record format
  Possible formats:
  - ☐ MicroSoft Word
  - ☑ WordPerfect
  - ☑ Xbase
  - ☐ MSAccess
  - ☐ Excel
  - ☐ Other
- ☐ Other (Please list)

_____

_____

What functionality is necessary for Performance Measure Calculation and Display?

- ☑ Custom performance measure interface
- ☑ Graphical screen display
- ☑ Generate report
- ☐ Export result in specialized record format
  Possible formats:
  - ☐ MicroSoft Word
  - ☑ WordPerfect
  - ☑ Xbase
  - ☐ MSAccess
  - ☐ Excel
  - ☐ Other
- ☐ Other (Please list)

_____

_____

What functionality is necessary for Data Storage and Retrieval?

- ☑ Multi year storage and retrieval
- ☑ Long term storage
- ☑ Security and encryption
- ☑ Store and retrieve data in specialized record formats
  Possible formats:
  - ☐ MicroSoft Word
  - ☑ WordPerfect
  - ☑ Xbase
  - ☐ MSAccess
  - ☐ Excel
  - ☐ Other
- ☑ Storage media
  Possible media:
  - ☐ Local Hard drive

- [x] Networked hard drive
- [ ] CD-ROM
- [ ] Tape
- [ ] Excel
- [ ] Other

- [ ] Other (Please list)

_____

_____

What functionality is necessary for Report Generation?

- [x] Standard reports
- [x] Custom-built report functionality
- [x] Print reports
- [x] Store reports/report database
- [x] User editing of reports
- [x] Report format

  Possible formats:
  - [ ] MicroSoft Word
  - [x] WordPerfect
  - [x] Xbase
  - [ ] MSAccess
  - [ ] Excel
  - [ ] Other
- [ ] Other (Please list)

_____

_____

What functionality is necessary for Decision Support/Modeling?

- [x] Optimization models
- [x] Network models
- [ ] Specialized, custom-built models
- [ ] Custom-built user interface to modeling
- [x] Interface to other modeling software

  Other modeling software:
  - [x] ARC/INFO
  - [ ] GAMS
  - [ ] TRANSCAD
  - [ ] AMPL
  - [ ] Other
- [ ] Other (Please list)

What other functionality must be included in the IMS?

- ☑ Electronic Data Interchange
- ☑ Fax/modem services
- ☐ Other (Please list)

_____

_____

What databases should be included in the IMS?

- ☑ Coal Handling Facilities
- ☑ Airports
- ☑ Transit Facilities
- ☑ Fertilizer Distribution Centers
- ☑ Grain Elevators
- ☑ Major Motor Carrier Terminals
- ☑ Sugar Beet Collection and Processing Facilities
- ☐ Other (Please list)

_____

_____

What performance measures should be included in the IMS?

Coal Handling Facility Database:

- ☑ Basic traffic characteristics
- ☐ Other (Please list)

_____

_____

Airport Database:

- ☑ Basic traffic characteristics
- ☐ Freight/Operations Efficiency Measure
- ☐ State Highway Location Metric
- ☑ Scheduled air service
- ☐ Other (Please list)

_____

_____

Grain Elevator Database:

- ☑ Basic traffic characteristics
- ☑ Modal access and share

- ☑ Rail shipment characteristics
- ☑ Truck shipment characteristics
- ☑ Elevator shipping capacity and annual volume
- ☐ Other (Please list)

Fertilizer Distribution Database:

- ☑ Basic traffic characteristics
- ☐ Car handling capacity
- ☑ Storage capacity
- ☐ Other (Please list)

Motor Carrier Database:

- ☑ Basic traffic characteristics
- ☑ Truck Type Unloading Time Metric
- ☐ Other (Please list)

Transit Facility Database:

- ☑ Basic traffic characteristics
- ☑ Operational characteristics
- ☐ Other (Please list)

Sugar Beet Collection and Processing Database:

- ☑ Basic traffic characteristics
- ☑ Other (Please list)
  Storage capacity

Systemwide:

- ☑ Commodity shipment routing characteristics, highway, rail and air
- ☑ Highway structural demands by freight shippers
- ☑ Support Pavement Management System (PMS)/Traffic Management System (TMS)
  - ☑ Compute intermodal average daily truck trips

- ☑ Identify commodities originated/terminated on specific highway sections
- ☑ Estimate intermodal facility generated truck traffic by highway section
- ☑ Project traffic demand with modal shift
- ☐ Other (Please list).

_____

_____

- ☐ Computer intermodal facility-specific ESALS
- ☐ Facility throughput
- ☐ Facility modal access
- ☑ Facility/Modal transfer time
- ☑ Service Frequency
- ☑ Waiting Time
- ☑ FRA Track Classification for rail access

What databases will be used together?

Grain and Fertilizer, check out others as we go along.

What databases will be used most often?

Grain Elevators, Sugar Beet Processing

What databases will be updated?

- ☑ Coal Handling Facilities
- ☑ Airports
- ☑ Transit Facilities
- ☑ Fertilizer Distribution Centers
- ☑ Grain Elevators
- ☑ Major Motor Carrier Terminals
- ☑ Sugar Beet Collection and Processing Facilities

When will these databases be updated?

- ☐ Daily
- ☐ Occasionally
- ☐ Annually
- ☐ Never

NDDOT Comment: Every 3-5 years.

How will these databases be updated?

- ❑ User input
- ❑ Outside file creation and installation
- ☑ EDI Updated files captured by EDI process.
- ❑ Interface to other software

  Please list:

  _____

  _____

Who will update the databases?

- ❑ User
- ❑ System Administrator
- ☑ Outside consultant

APPENDIX B

IMS COMPUTER USE SURVEY

Intermodal Transportation Management System Computer Use Survey

SURVEY RESULTS

37 Surveys were sent out. 30 responses were received. (81%)

Organization: *Department of Transportation, Planning Division*

Directions: Please answer the following questions carefully. Please circle your answer or fill in the blank.

SECTION I. Computer Use

			Results	% of Respondents
1.	What kind of computer do you use?			
	a.	PC	30	100
	b.	Work Station	2	7
	c.	Mainframe	4	14
2.	How many years have you used a computer?			
	a.	1	3	10
	b.	3	3	10
	c.	5	5	17
	d.	more than 5	19	63
3.	What operating system do you use?			
	a.	DOS	22	76
	b.	Windows	15	50
	c.	OS/2	1	3
	d.	Other	4	14
4.	Approximately how many hours a week do you use your computer?			
	a.	10 or less	6	20
	b.	10 - 20	12	40
	c.	over 20	12	40

5. Please circle the input devices do you use.

Keyboard	30	100
Mouse	21	70
Pen	1	3
Trackball	0	0
Other	3	10

6. Are you on a network?

Yes	5	15
No	25	85

7. Please circle the computer software programs you use or are familiar with.

Word processor	23	76
Database	22	73
Statistical package	2	6
Spreadsheet	18	60
Graphics programs	13	43
Application Generators	1	3
Programming Languages	1	3
CAD	10	33
Decision Support Systems	1	3
Desktop Publishing	3	10
Multi-Media	0	0
Networking	1	3

8. Do you use software on a network?

Yes	4	13
No	26	87

9. Do you develop computer programs or information systems?

Yes	3	10
No	25	83

SECTION II. Graphical User Interfaces (GUI)

10. Have you used a Graphical User Interface such as Windows?

Yes	17	55
No	13	45

11. Please list the Graphical User Interfaces you have used.

_____

_____

Windows	11	33
Autocad	1	3
ACAD	1	3
Freelance	1	3
Wordperfect for Windows	1	3
OS/2	1	3
Mouse Pointer	1	3
Tranplan	1	3
Dosshell	1	3

12. Do you use a Graphical User Interface in your work?

Yes	16	53(60)
No	12	40(40)

13. If you use Graphical User Interfaces, how helpful are they?

a.	a little	1	3(6)
b.	somewhat	3	10(19)
c.	a lot	12	40(75)

14. Please circle which GUI features you are familiar with.

pull down menu	16	53
pop up menu	10	33
dialog box	8	27
scroll bar	7	23
menu bar	17	57
list box	7	23
window	17	57
icon	16	53
message box	8	27
input box	7	23
command button	12	40
mouse pointer	16	53

SECTION III. Information Use

The following questions are used to learn how you use information and data in your work.

15. What type of information do you use in your work?

Raw data	26	87
Statistics	14	47
Written Reports	19	63
Graphs	18	60
Charts	13	43
Maps	19	63

16. In your work, do you read and use information directly from a computer screen?

Yes	24	80
No	6	20

17. Approximately how many hours a week do you use your computer to create or retrieve information?

a.	10 or less	11	38(38)
b.	10 - 20	12	40(41)
c.	over 20	6	20(21)

18. Do you use any computerized information systems like the Pavement Management System?

Yes	12	40
No	18	60

19. Do you create or analyze data for others to use in their jobs?

Yes	14	47(48)
No	15	50(52)

20. Please describe the type of information you find most useful in your work.

_____

_____

Traffic Count
Accident Data
Transportation Trend Data
Work Schedules
HPMS
RIMS

Budget Reports
Personnel Information
Man hours per job
Graphical info such as maps tied to databases
All roadway info,r/w,utilities,traffic
Mile by mile Raw Data
Pavement Management Information,traffic data,truck data
Pavement Performance Data, Project Data
Data
Graphs
Transportation planning related
I do almost strictly word processing
Economic factors & trends affecting local gov't.
The storage and retrieval of all my work,data, and information in my PC
Existing roadway data

21. Please circle all the ways you use information in your work?

Analysis	22	73
Decision-making	20	66
Writing reports	20	66
Reports for presentations	22	73
Making charts	17	57

22. Do you work as part of a team that directly uses the same information?

Yes	23	77
No	7	23

23. Do you think computerized information systems useful?

Yes	30	100
No	0	0

SECTION IV. Intermodal Transportation

24.	How much experience do you have with Intermodal Transportation?			
	a.	none	7	23(25)
	b.	a little	11	37(39)
	c.	some	9	30(30)
	d.	a lot	1	3(4)
25.	Do you need to consider Intermodal Transportation in your job?			
	a.	None	3	10(12)
	b.	a little	6	20(23)
	c.	some	14	47(54)
	d.	a lot	3	10(12)
26.	Does Intermodal Transportation affect your department?			
	a.	none	0	0
	b.	a little	0	0
	c.	some	10	33(37)
	d.	a lot	17	57(63)
27.	Will Intermodal Transportation become more or less important?			
		More	26	87(100)
		Less	0	0

Comments:

Intermodal transportation needs to be defined.
I see the use of laptops(notebook) computers becoming more and more the wave of the future. Especially for traveling and documenting meetings, field trips, and public hearings.
Intermodal information is often unavailable for use in analysis of highway improvements.
Truck, train, and pipeline commodity movement and the interface between the modes should be considered in the design of highways as well as the other modes.
Currently work with Metro Planning Organizations and smaller urban areas: To develop transportation plans (..?.. 20 years). Also special area studies and individual area studies.
Yes because of the requirements of ISTEA.
I believe our transportation systems are just fine, or at least as good as

any in the Midwest. A lot of the urgently expressed needs are somewhat overdramatized. Considering our pop.,land area, tax base, etc. we're doing all-right in N.Dak.
I basically work in a supervisory/administrative role whereby I direct development of highway improvement projects in cities. This gets into intermodal issues. I also provide assistance/admin to Metropolitan Planning Organizations (IE FMCOG) & the urban areas of North Dakota in areas of urban planning, transit planning, & bicycle/pedestrian planning. Although my staff use data for analysis on the PC, I do very little data analysis myself. Data includes: VMT(vehicle miles traveled) on a street system, ADT(average daily traffic), accident data & stats, PMS data, as well as social/economic data for use in developing traffic projections.
