THE VISUALIZATION OF OBJECT-ORIENTED ENTERPRISE MODELING

by

XU ZHANG

B.Sc., Zhejiang University, 1992

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN BUSINESS ADMINISTRATION

in

THE FACULTY OF GRADUATE STUDIES

(Faculty of Commerce and Business Administration)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

October 1998

© Xu Zhang, 1998
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Faculty of Commerce & Business Admin.

The University of British Columbia
Vancouver, Canada

Date Oct. 29, 1998
ABSTRACT

This research work is focused on the development of a visual Computer-Aided Software Engineering (CASE) tool for Object-Oriented Enterprise Modeling (OOEM) methodology. Based on an ontological foundation, OOEM is an object-oriented analysis and design methodology that provides a set of rules for and a systematic approach to enterprise modeling. Previous research has demonstrated that the ontological rules of OOEM could be stored within a CASE tool to direct model construction and check model integrity. The thesis designs and technically implements such a CASE tool for OOEM. It explores theoretical and technical solutions to the issues identified by previous research; investigates more OOEM modeling and analysis features that could be incorporated into a visual CASE tool; and also studies empirically the usefulness and accessibility of such a tool to its intended user audience. The empirical study shows that the resulted CASE tool does help analysts to appreciate OOEM strengths and to apply the methodology with increased effectiveness and efficiency.
Table of Contents

ABSTRACT ........................................................................................................................................ II

TABLE OF CONTENTS ......................................................................................................................... III

LIST OF TABLES ................................................................................................................................. VII

LIST OF FIGURES ............................................................................................................................... VIII

ACKNOWLEDGEMENTS ..................................................................................................................... IX

1 INTRODUCTION ..................................................................................................................... 1
  1.1 THESIS OBJECTIVES .............................................................................................................. 2
  1.2 THESIS OUTLINE .................................................................................................................. 3

2 BACKGROUND ............................................................................................................................... 4
  2.1 OBJECT-ORIENTED MODELING AND ANALYSIS .............................................................. 4
  2.2 OBJECT-ORIENTED ENTERPRISE MODELING (OOEM) ...................................................... 5
    2.2.1 Ontological Foundation ...................................................................................................... 6
    2.2.2 Modeling Constructs .......................................................................................................... 7
      2.2.2.1 Objects ....................................................................................................................... 7
      2.2.2.2 Requests .................................................................................................................... 8
      2.2.2.3 Services .................................................................................................................... 8
      2.2.2.4 Attributes ................................................................................................................ 9
      2.2.2.5 Systems .................................................................................................................... 9
    2.2.3 Modeling Rules .................................................................................................................. 10
      2.2.3.1 Rule #1: The Scope Identification Rule ..................................................................... 10
      2.2.3.2 Rule #2: The Object Identification Rule .................................................................. 10
      2.2.3.3 Rule #3: The Service Inclusion Rule ....................................................................... 11
List of Tables

Table 4.1 The Object Template.......................................................... 35
Table 5.1 OOEM Transformation Relationship...................................... 51
Table 6.1 Summary of Questions for User Friendliness........................ 71
Table 6.2 Summary of Questions for Modeling Capability....................... 73
Table 6.3 Summary of Questions for Analysis Capability....................... 76
Table 6.4 Summary of Questions for the System in General................. 78
List of Figures

Figure 4.1 General System Structure ................................................................. 29
Figure 4.2 OOEM Object Communication Diagram ............................................ 33
Figure 4.3 Perspective Diagram ....................................................................... 37
Figure 4.4 Object Communication Diagram for Figure 4.3 ............................... 39
Figure 4.5 Subsystem with No Office Manager from Figure 4.4 ....................... 41
Figure 4.6 Subsystem with No Request for Contract Review from Figure 4.4 ..... 43
Figure 5.1 OOEM Metamodel ........................................................................... 49
Figure 5.2 ER Diagram for OOEM CASE Tool Design ................................. 52
Figure 5.3 Algorithm for Object Communication View ................................... 54
Figure 5.4 Algorithm for Perspective View ...................................................... 55
Figure 5.5 Algorithm for Subsystem View ....................................................... 57
Figure 5.6 Sample Request Representation and Optimization ......................... 63
Acknowledgements

This research work is supervised by Dr. Carson Woo and Dr. Yair Wand of the University of British Columbia. Drs. Wand and Woo are proponents of Object-Oriented Enterprise Modeling (OOEM) methodology. I am very grateful for their invaluable suggestions, comments, and other support throughout the whole research period.

I also want to thank all the volunteer participants in the empirical study. Their time and special contributions to this thesis in regard to CASE tool evaluation are greatly appreciated. Their comments are important not only to this project but also to any future research.

Finally, I should give thanks to my wife Jin for her encouragement and caring. Without her support, I would have needed far more effort and time to finish the research.
1 Introduction

Many methods and techniques exist for object-oriented analysis and design. Most of these methods evolved from object-oriented programming languages. They were oriented towards software developments and, therefore, lacked rules and principles for enterprise modeling. Embley et al. (1995, p19) state that "many so called analysis methods are really preliminary design methods.... Many of their features are more appropriate for design than analysis."

Accompanying this fundamental weakness in current methods is an absence of software tools that provide a rule-driven approach to enterprise modeling. A survey of current computer-aided software engineering (CASE) tools for object-oriented analysis reveals the absence of software tools that provide rules for model construction and model validation (Tan 1997, p6).

Researchers have generated solutions to these weaknesses of current object-oriented modeling methods. One solution is the Object-Oriented Enterprise Modeling (OOEM) methodology developed by Wand and Woo (1993). Their methodology provides a set of clearly defined semantic rules based on ontological principles. These rules offer users step by step guidelines for the modeling of a problem domain. They can also be used to validate the semantic integrity of enterprise models. An OOEM CASE tool (OOEMCT) based on this methodology has demonstrated its usefulness (Tan 1997).
1.1 Thesis Objectives

Based on the set of OOEM rules, OOEMCT has become an object-oriented modeling tool that is able to provide a formal algorithm for enterprise modeling. Moreover, the built-in semantic engine of OOEMCT can evaluate candidate models against well-defined OOEM rules.

However, as a pilot research project, OOEMCT still has a number of serious shortcomings. These shortcomings prevent OOEMCT from gaining wide user acceptance. Lack of model visualization is a major drawback.

This thesis is motivated by the desire to effectively use the potential power of such a CASE tool. Although initially addressing the lack of model visualization and other shortcomings of OOEMCT, this research is expected to achieve more results. The author takes the view that it is possible, and indeed important, to have a real CASE tool for OOEM that supports its modeling rules, extends the power of its principles, and thus positions itself uniquely on the software market. Therefore, the objectives of this project are:

- To explore theoretical and technical solutions to the shortcomings associated with OOEMCT.
- To investigate more OOEM features that could be incorporated into a visual modeling tool.
• To test the usefulness and accessibility of such a tool to its intended user audience.

1.2 Thesis Outline

The forthcoming chapters present the research work aimed at achieving the above objectives.

Chapter 2 explains the fundamental concepts upon which the modeling tool was built. Notably, the chapter introduces the concepts of object-oriented modeling and analysis in general as well as, the new object-oriented enterprise modeling methodology, and computer-aided software engineering. Chapter 3 analyzes the objectives that the modeling tool is expected to achieve, along with the modeling tool’s requirements. Chapter 4 defines the functional interfaces of the proposed system. Chapter 5 provides the architectural design and implementation algorithms. Chapter 6 discusses the results of an empirical user study of the modeling tool. Finally, chapter 7 outlines future research work and concludes the thesis.
2 Background

This chapter is intended to provide background knowledge for those new to object-oriented analysis or to OOEM methodology. There is also a short introduction to computer-aided software engineering at the end of this chapter. The introduction is aimed to provide basic software engineering concepts that are used in later chapters.

2.1 Object-Oriented Modeling and Analysis

People use models to capture the aspects of a problem domain. A model is defined as

"an abstract representation of reality that excludes much of the world’s infinite detail. The purpose of a model is to reduce the complexity of understanding or interacting with a phenomenon by eliminating the detail that does not influence its relevant behavior. Therefore a model reveals what its creator believes is important in understanding or predicting the phenomena modeled" (Curtis 1992).

Analysis is the study and modeling of a system’s behavior and characteristics. Traditional systems analysis methods focused on soliciting the requirements for a software design. This software engineering focus has continued in the development of object-oriented analysis methods.
Object-oriented analysis (OOA) is an approach to modeling a problem domain using object-oriented constructs. It originated from object-oriented programming technology. With the proliferation of proposed OOA methods, the term object has been defined in many ways. However, there is an implicit basic agreement that all objects embody an abstraction. For instance, Rumbaugh, et al. define an object as "a concept, abstraction, or thing with crisp boundaries and meaningful for the problem at hand" (Rumbaugh, et al. 1991, p21).

To better explain the OOA approaches, a well-known OOA method provided by Coad and Yourdon (1991) is useful as a general example. The Coad and Yourdon method is based on a five-layer model. The layers, from top to bottom, are the object layer, the structure layer, the service layer, the attribute layer and the subject layer. The object layer denotes the objects. The structure layer captures various structures among objects, e.g. one-to-many relationships. The service layer denotes messages and object behaviors. The attribute layer details the attributes of objects. Finally, the subject layer divides the design into implementation units or team assignments. These layers provide the structure for building an OOA model through five primary activities: finding objects, identifying structures, defining services, defining attributes, and identifying subjects.

2.2 Object-Oriented Enterprise Modeling (OOEM)

Traditional OOA methods are used to gather the software requirements for system implementation. The OOEM approach takes a different viewpoint. Instead of modeling
from the perspective of software development, OOEM is used to develop models at the enterprise level. OOEM defines the term *Enterprise Model* as “a symbolic term which captures the essential and abstract relations and properties, both structural and dynamic, of an organization, usually as a whole” (Zhao 1995, p5). In enterprise modeling, analysts shift emphasis from modeling software requirements to modeling organizational activities, business policies, and business models.

2.2.1 Ontological Foundation

Unlike other object-oriented analysis techniques that lack a strong theoretical foundation, OOEM methodology is grounded on ontological principles.

Ontology, a branch of philosophy (metaphysics), is the study of theories of being that involves the basic traits of all reality. Bunge states, “Metaphysics studies the generic (nonspecific) traits of every mode of being and becoming as well as the peculiar features of the major genera of existents” (1977, p.5); “Metaphysics can render service by analyzing fashionable but obscure notions such as those of a system, hierarchy, structure, event, information…” (1977, p.24). Such an ontological approach provides guidelines on how to model the world.

Because an enterprise model captures information about organizational *things*, such as customers, products, employees, and orders, a theory of the nature of things can function as the foundation for enterprise modeling (Jung 1997, p15).
The following is a summary of the ontological principles that serve as a basis for enterprise modeling (Zhao, 1995):

- The world is composed of things.
- Things have properties.
- Every thing abides by laws that are invariant relations among properties of the thing.
- Every thing changes and every change is a change of things.
- Interacting things form systems or aggregates.

2.2.2 Modeling Constructs

Using the ontological principles, ontological constructs can be mapped onto object-oriented constructs. The latter constructs allow us to represent and reason about the problem domain using the object-oriented concepts.

This section introduces each of these object-oriented constructs, together with their ontological meanings.

2.2.2.1 Objects
Ontologically, a thing is considered the basic unit of existence. Accordingly, OOEM asserts that a model of the world is made of objects. The object is the unit of analysis rooted in the principle that the world is composed of things. Zhao (1995, p.12) defines an OOEM object as “a model of substantial thing that interacts with other things (modeled as objects) in the problem domain.”

Objects can be classified as internal and external based on the scope of the problem domain. Internal objects are considered part of the domain. External objects are objects that interact with the domain but are not considered part of it.

2.2.2.2 Requests

An object interacts with other objects by sending requests. Requests model the interactions among objects. Ontologically, interaction is the change of state of a thing as a result of the existence of another thing. A request can change the state of the sender, the recipient, or both.

Requests are categorized into two types: external requests and internal requests. External requests originate from external objects. Internal requests reflect the interaction of internal objects.

2.2.2.3 Services
Services represent the possible state transitions of a thing. A service is a series of well-defined actions taken by an object to satisfy a request. A service, in its course of action, may access or modify the state of the object itself, or generate one or more requests to other objects.

2.2.2.4 Attributes

Objects have attributes as things have properties. An attribute is a characteristic assigned to an object that describes the state of it. Attributes also represent the object's knowledge of the problem domain.

OOEM divides attributes into two types: internal attributes and interface attributes. Other objects do not publicly know internal attributes. They can only be accessed or changed through the services of their owner objects. Interface attributes are the means for object communication and are publicly accessible by others.

2.2.2.5 Systems

A group of interacting objects form an aggregate known as a system. A system models a real world system that is composed of interacting things. External objects model the system environment. A request from an external object causes a state change in an internal object and changes both the system and affected object into an unstable state. A
set of state transformations will then happen within the system until the system reaches another stable state.

2.2.3 Modeling Rules

The OOEM modeling rules provide the core principles for creating and using the object-oriented constructs. There are a total of seven ontologically based rules that define the course of modeling a domain (Wand and Woo 1996). While the first five rules tackle system dynamics, the other two rules are concerned primarily with the structural aspect of the system.

2.2.3.1 Rule #1: The Scope Identification Rule

This rule determines the model's scope by stating that the analyst should define the system's boundary in terms of the activities to be supported by the problem domain. The rule distinguishes between the environment and the enterprise; the environment is represented by external objects. These external objects interact with the modeled system by sending requests to it, or receiving requests from it.

2.2.3.2 Rule #2: The Object Identification Rule

This rule determines which real world things should be modeled as objects. It states that an object is included if and only if it provides at least one service, or makes at least one
request to the system. An internal object is part of the system and provides at least one service. An external object is part of the environment that interacts with the system.

### 2.2.3.3 Rule #3: The Service Inclusion Rule

This rule determines the services to be modeled. It states that a service is included in the system if and only if it is directly invoked by at least one request. This rule does not prevent a service from being invoked by multiple requests.

### 2.2.3.4 Rule #4: The Attribute Inclusion Rule

The rule determines which attributes should be included in a model. To be necessarily included, an attribute must be either used or affected by at least one service, or known to at least one other object. An internal attribute is different from an interface attribute in that it is invisible from outside.

### 2.2.3.5 Rule #5: The Attribute Ownership Rule

The rule states that an attribute belongs to one and only one object. This object is known as the *custodian* of the attribute. The value of every attribute can only be modified or obtained by the services of its custodian object.
2.2.3.6 Rule #6: The Composite Object Rule

This rule determines when to include composites or component objects in the model. Here the term *composite* is used to refer to an aggregate object that can be decomposed into component objects in the problem domain.

The Composite Object Rule states that a composite may only be included if it provides services that are not provided by any of its component objects. Only properties that are not properties of components should be included in a composite object.

2.2.3.7 Rule #7: The Sub-Classification Rule

This rule determines when to create a general object class. This general object class (super-class) has the common services that are provided by two or more objects (sub-classes). All common services provided by the super-class should be eliminated from the sub-classes. Alternatively, one can form a sub-class only if it includes new services and attributes with respect to the super-class.

The Sub-Classification Rule represents the model's generalization and specialization structure.

2.2.4 Request Propagation
Wand and Woo’s methodology provides not only a set of rules but a request propagation algorithm that systematically apply the rules to build a model.

The general idea of request propagation came from the ontological law for state transformation. As stated previously, an external event may stimulate a real world system into state transformation. This transformation would propagate throughout the system until a stable state is reached.

OOEM shares the same belief that a system is triggered by external requests. These requests interact with internal objects and invoke system services. In the process of performing a service, an object may generate requests to other objects. This request propagation only ends with external objects, or objects that do not need to propagate requests.

Previous research proves that, by tracing this request propagation process, the analyst can identify all objects and their associated services, interface attributes, internal attributes and connected requests (Zhao 1995, p36). A system model can be produced if all the above information is present.

2.3 Computer-Aided Software Engineering
As early as 1972, Bauer defined software engineering as “the establishment and use of sound engineering principles in order to obtain, economically, software that is reliable and works on real machines”

The concept of software engineering has evolved since then. A formal Systems Development Life Cycle (SDLC) is now widely accepted by software engineers. The SDLC is initialized by project opportunities. After that, it goes through system planning, system analysis, system design, system implementation, and finally system maintenance.

Computer-aided software engineering (CASE) tools integrate with SDLC by providing an automated framework for it. Software engineers have benefited from these productivity tools created explicitly to improve their routine work through the use of programmed support. The CASE tools also improve communication between analysts and users. Clients can see various useful diagrams and reports generated by the computer.

CASE tools are usually classified as front-end or back-end, according to the development phase(s) which they are meant to support. A front-end CASE tool supports the planning, analysis, and design stages of SDLC, while a back-end one provides support for coding and implementation.

They are plenty of front-end CASE tools available in the software market. The next sections introduce two that specialize in object-oriented modeling.
2.3.1 Rational Rose

Rational Rose is one of the leading visual modeling tools. It supports object-oriented analysis, modeling, and design with the Unified Modeling Language (UML). A system analyst can use the tool to generate Use Case diagrams, Class diagrams, Sequence diagrams and more. These diagrams complete the UML notations and provide a basis for software development (Quatrani and Booch 1998).

A Use Case of UML models how external actors make use of the system. It delimits the modeled system's boundaries and interactions with the environment. Rational Rose visualizes this modeling process through its Use Case diagrams.

The Class diagrams show the existence of classes and their relationships in a logical view of a system. These diagrams are also used to capture the attributes and services of the system.

The Sequence diagrams are used to trace the execution of different activity scenarios within a system. They model object interactions in terms of the order of requests and services.

The visual diagrams of Rational Rose graphically represent the structure and interrelationships of a system. They make system design and development well organized, much clearer, and easier to maintain.
However, UML and Rational Rose only provide a set of object-oriented modeling notations from the viewpoint of software development. They do not support for a method or process for using the notation to create object models. Meanwhile, no intuitive and smooth connections exist among different models. For example, a user may create a model in a Use Case View, but he/she can not make any connections between the Use Case model and the Class model. In addition, the CASE tool does not present decomposition in a hierarchical manner.

To sum up, Rational Rose may be a good CASE tool for software design and development, but it does not adequately address the enterprise level of modeling.

2.3.2 OOEMCT

OOEMCT (Tan 1997) is a form-based CASE tool that supports OOEM methodology. It provides two ways of modeling building, named Guided mode and Free Form mode.

In the Guided mode, analysts are led through a predefined sequence of forms that will enable them to construct the model in a structured way. This approach follows the request propagation algorithm of the underlying methodology.

The Free Form mode allows analysts to directly enter model constructs and their relationships. This approach provides more flexibility to experienced users.
The CASE tool is also integrated with a diagnostic report. The report can be run against the OOEM rules to identify those semantic inconsistencies that may exist in a model.

Value-added by the semantic checking capability, OOEMCT is the first CASE tool that provides systematic object-oriented enterprise modeling. However, with no diagrammatic model representations, OOEMCT users generally find it difficult to generate and follow their modeling conceptions.
3 Requirements Analysis

This chapter analyses the objective and requirements of the proposed system from the perspective of system analysts.

3.1 The Objective

A software product must be useful. “It is not our job to produce clever computer applications that reduce productivity” (Landauer 1995, p. 663). Moreover, we are developing a tool dedicated to the OOEM methodology. The product should be customized to support modeling with OOEM. It is not our purpose to develop a generic CASE tool, no matter how useful it might be. Taking these two points into account, the author’s application objective is to help the analyst effectively and efficiently model business domain using OOEM methodology.

Effectiveness measures how well the model represents important facts of the domain. Efficiency measures the resources required to model the domain (Wand 1993, p.2). A good CASE tool should increase both the effectiveness and efficiency of the underlying methodology.

3.2 System Requirements
The purpose of requirements analysis is to determine what the analyst expects from this modeling tool. Basically, the requirements should be consistent with the system objective as stated in the previous section.

The author consolidates the system requirements from a number of resources, including the following:

1. Shortcomings associated with OOEMCT.
2. OOEM methodology and its implications.
3. Features provided by other object-oriented modeling tools.
4. The findings of a user study conducted by Tan (1997, p67-86) for OOEMCT.
5. Consultations with system analysts concerning their expectations for the modeling tool.
6. The author’s own experience in the areas of software engineering and system modeling.

Tan (1997, p88) identified three major obstacles that prevent OOEMCT from gaining total user acceptance. This modeling product is intended to remove these obstacles. However, this is only one of many requirements. The software is expected to fully utilize the strengths of the underlying methodology while addressing common modeling needs as other CASE tools do.
The system requirements are divided into groups that address different user needs, namely user friendliness, domain modeling, and system analysis.

3.2.1 Solving Problems

This sub-section introduces solutions to existing OOEMCT problems. The proposed solutions actually fall into other broader ranges of system requirements. However, they are grouped together since the main motivation of this research project is to solve these problems.

3.2.1.1 Visualization

Users of OOEMCT have found the absence of a graphical user interface (GUI) to be a major stumbling block (Tan 1997, p88). This is not surprising. With the emergence of Windows and visual tools, the expectations of software users have been raised. Now, they are used to various kinds of diagrams and mouse drag-and-drops.

GUI is not the only component of computer visualization. With this modeling tool in particular, visualization plays two more important roles: model construction and model presentation.

3.2.1.1.1 Model Construction
Graphical model construction enables a user to directly manipulate a model on the screen, thereby reducing the number of mental processing operations for the user: "Mental processing operations include requirements for the user to learn complex commands and syntax, memorize encrypted codes and abbreviations, or translate data into other units or formats before they can be applied to the problem at hand" (Brown 1988, p4).

By having a graphic canvas for model structuring, the analyst can focus more on modeling the problem domain itself, thereby saving the analyst significant time and effort which would otherwise be necessary to memorize and mentally process the operations inherent in the design of system units and their various linkages.

3.2.1.2 Model Representation

Visual model representation offers the analyst cognitive advantages. The human ability to extract information from visual scenes is much more fundamental than is our ability to manipulate data verbally or arithmetically (Schwartz and Howell 1985).

Model diagrams enable the analyst to extract, process, understand, and respond to much relevant model information. This transfer of information is fast, accurate, and the user learning curve is minimized; the analyst can thus build a conceptual model of the problem domain with fewer perceptual errors (Brown 1988).
The diagrams can also serve as the interface between a domain analyst with his/her customers. The visual model representation is indeed worth thousands of words in terms of communicating with customers.

3.2.1.2 Object Typing

The type of OOEMCT object can not be changed during model construction. An object first created as an external object will always be external and send requests to the system. Similarly, an internal object will remain internal.

The new modeling tool breaks this restriction by allowing an analyst to change the object type at any stage of model construction. This flexibility offers two main benefits.

1. The boundary between the system (internal objects) and system environment (external objects) is drawn by the analyst. Such a boundary reflects the analyst's view of reality; his/her understanding of the portion of reality that is of interest to him/her (Zhao 1995, p23). This portion could be extended or narrowed when the modeling scope is changed or the problem domain itself changes.

2. An empirical study shows that analysts prefer to defer the decision on object typing until they are ready (Tan 1997, p90).
The ability to switch between object types leaves the analyst more space for restructuring ideas without loss of useful information. It also adds great flexibility to capturing system dynamics.

3.2.1.3 Diagnostic Feedback

The OOEMCT diagnostic report lists the model’s inconsistencies with OOEM rules. However, the report is text based with no link to the problem constructs. Users found it inconvenient for error tracing. Besides, reported errors might not be corrected if they were found too late (Tan 1997, p90).

This problem is associated with the previous two ones. The lack of model visualization reduces the expressive power of the semantic report. Model mistakes are also hard to be fixed if they are related to objects and their types. As stated before, the tool does not allow an object’s type to be changed.

To solve the diagnostic feedback problem, we need to redesign the report interface so that the users can easily identify and fix model inconsistencies.

3.2.2 User Friendliness

User friendliness is a system requirement that cannot be overemphasized. Humans often tend to resist change. Any new piece of software changes some habits people have
become used to. Applications that are not easy to use make user acceptance even harder. Remarkable systems have been discarded by users due to these systems’ lack of user friendliness (Brown 1988).

The concept of user friendliness should not be limited to interface design. A well-designed user interface is only the first step towards a user friendly system. A user friendly system also implies good designs for both novice and expert users, and the ability to manage complexity.

3.2.2.1 Designs for Novices, Experts, and Intermediate Users

The system should be designed to incorporate the needs of novices, experts, and intermediate analysts of OOEM. Ignoring any of them would greatly restrict the user base of this modeling tool. A novice model builder becomes experienced sooner or later, moreover, few users are experts in all aspects of enterprise modeling.

Novices may need extensive guides and prompts. As a result, the system must provide a guided mode for model construction, based on the rule-driven methodology and request propagation algorithm of OOEM.

Experts should be able to shortcut or bypass the ordered action sequences. A relatively simple method, or ‘free’ mode, should be provided to them to capture model essentials.
The shift between the guided mode and the free mode should be smooth enough to address the needs of intermediate users. Novice users could be intermediate in some modeling areas, as could experts as well.

### 3.2.2.2 Complexity Management

An enterprise model may have tens or hundreds of objects. It is impossible to crowd them all together. The best way to manage this model complexity is to add object layers. The idea of object layers is supported by the Object Composite Rule of OOEM.

The Object Composite Rule of OOEM states that a composite object can be decomposed into component objects in the problem domain. By doing so, an enterprise model can be structured into layers. Each low-level layer includes model components that belong to a composite object of the upper layer. For example, a University may be a composite with the Registrar's Office and Faculties as components.

When decomposing, the composite object will be completely replaced by the more detailed description of components and their interactions. The rule governing this decomposition is that every request and response going in and coming out should be manifested at the lower level. The only exception to this rule states that a request-response pair can be broken into two separate requests, one incoming and the other outgoing, but at a lower level.
This exception reflects the fact that more knowledge can be expressed with more details. For example, at a high level, a student applies to and receives a response from a University. At a lower level, a student applies to the Registrar's office and receives a response from a Faculty.

3.2.3 Modeling Capability

The modeling capability of the system determines the usefulness of this product. As a modeling tool, the system is required to support the entire OOEM modeling process, a process that can be divided into stages.

In the initial stages, the tool should help analysts to gather information on an organization's activities, to discover and collect ideas, to structure them, and finally to put them together into a comprehensible model. This process is known as idea structuring (Card and Moran, 1995).

After that, the tool should apply the OOEM rules to evaluate the constructed model. The semantic inconsistencies between the model and problem domain trigger a further round of idea structuring and restructuring until the model is internally correct or until the analyst chooses to finish it.

The tool should also allow the analyst to suspend model construction at any stage and resume it at a later time. No information should be lost between any of these actions.
3.2.4 Analysis Capability

Recall that the purpose of enterprise modeling is not only to automate the existing processes, but make them better structured and more productive. Modeling is only the first task a system analyst needs to perform. To make improvements is essentially the next step.

After the business activities are captured and represented by an OOEM model, the tool is expected to support the follow-up system analysis against the model. The analyst may manipulate the model to play various scenarios and answer different what-if questions. This process leads to the development of possible improvements to the existing processes.

The analysis capability of the CASE tool not only adds more power to this modeling software but also makes it more interesting to use.
4 System Approach

This chapter is intended to blueprint the new OOEM modeling tool. It covers the functional interfaces of the proposed system with external users. Concepts and reasons underlying the design are discussed, while examples are given to further explain the approaches.

4.1 General Functionality

To fulfill user expectations and system requirements, the program is designed with six major interfaces. An empirical study has been conducted to test how well this system approach matches the product's objective and requirements. A detailed discussion of this empirical study is in Chapter 6.

Figure 4.1 shows the general system structure. The diagram shows a data repository that sits in the center. Other modules surround it and communicate with it. The communication is represented by arrows. A double arrow means both access and update. A single arrow indicates read-only. The data repository functions as a central place for information sharing. No direct data exchange occurs between the other components.
The following sections introduce the purposes and concepts of these top-level system components, starting with the data repository.

4.2 Data Repository

The data repository is one of the most important components of the modeling tool. It keeps track of the current OOEM model created by the systems analyst. Information stored in the data repository can be categorized into the following three areas: model information, graphic information and system information.
1. Model information describes the objects, requests, services and attributes within the system. It also describes the relationships among these constructs. This information itself describes the OOEM model.

2. Graphic information is the topological information required by structured diagrams. It captures the position and size of graphical components. These graphical components represent the visualization of OOEM modeling constructs.

3. System information is used to trace the historical and current user behaviors; e.g. number of diagrams created, descriptions of subsystems, and user options. Such information is supplementary yet important because it saves the user's time that would otherwise be needed to reproduce the scenarios.

Since all user operations are finalized as operations on the data repository, an efficient and sound database engine is vital. The author chose a relational database structure for the data repository for two reasons. Firstly, the well-structured SQL language provides a powerful yet flexible query capability. Secondly, given the dominance of relational databases in the database market, handy database management tools are available to choose from.

Given the nature of enterprise modeling, the data repository has the following properties: limited number of data entries, comprehensive data relationships, and easy transaction
management. These properties can be used as guidelines for database design and for choosing the right database management tool.

1. The OOEM is used to model organizational activities. Since there is a finite number of activities within any organization, the number of model constructs (data entries in the data repository) will not be significant. This property has two implications for database selection and design. Firstly, database index is not necessary and, secondly, database denormalization can not significantly increase system performance (Rob 1995, p210).

2. The data repository stores not only model constructs but also the relationships among constructs. These relationships are built upon the seven rules of OOEM. For example, the Attribute Ownership rule states that an attribute must be owned by an object. These relationships are defined by database links referred to as referential integrity. Maintaining referential integrity is a key component of repository management.

3. The current version of the modeling tool is not designed to support concurrent users. This eliminates the need for concurrency controls and simplifies database transaction management.

The next chapter gives a detailed description of data repository designs and referential enforcement rules.
4.3 Object Communication Generator

Object Communication Generator visualizes the OOEM model. It also allows the analyst to freely identify objects, their services, attributes, and interactions that occur among the objects. There are two complementary types of components within the Object Communication Generator: the object communication diagram (OCD) and the object template (OT). The combination of an object communication diagram and object templates adequately capture the OOEM essentials.

4.3.1 The Object Communication Diagram (OCD)

The object communication diagram captures and communicates the scope of the modeled system, including the following sets of elements:

- the set of external objects
- the set of internal objects
- the set of requests that show object interactions
- the set of services
- the set of attributes
Figure 4.2 depicts an example of a generic OCD. The OCD identifies the scope or boundary of the system by distinguishing external objects from internal objects. The solid arrows represent the flow of requests for services from one object to another. Services and attributes are shown under their custodian objects. OCD provides the option of showing or not showing them.

Legend

<table>
<thead>
<tr>
<th>Internal Object</th>
<th>External Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Name</td>
<td>Object Names</td>
</tr>
<tr>
<td>Services</td>
<td>Request</td>
</tr>
<tr>
<td>(This section is optional)</td>
<td></td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
</tr>
<tr>
<td>(This section is optional)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.2 OOEM Object Communication Diagram
The object communication diagram concentrates on depicting the interactions among objects while hiding object details. It shows a clear picture of interactions within the organization.

4.3.2 The Object Template (OT)

The object template is used to display and edit key characteristics for both internal and external objects (Zhao 1995, p33). An object template is invoked by clicking the selected object on the object communication diagram.

Table 4.1 gives the structure of OT. Within the template, an object is defined by the object name in the header field. A set of services that the object needs to perform is listed in the Services column. The Triggering Requests column records the corresponding requests that invoke the services.

The next two columns show the interface and internal attributes as required by the object in the course of providing a service. There are two types of interface attributes. Incoming interface attributes refer to the information passed along with a request. Returning interface attributes are the responses to a request upon completion of the invoked service.

In the process of providing a service, an object may have to request services from other objects. These requests are recorded under the Requests Generated column. For an internal object, the Incoming and Outgoing Requests are the unions of all triggering and
generated requests, respectively. For an external object, Incoming Requests are requests for services from the system, and Outgoing Requests are requests sent to the system.

The last column stores all the custodian attributes of the object. It is a union of interface and internal attributes by all the services.

Apparently, the last three columns store redundant information that can be derived from other columns; this design actually achieves extra flexibility for model construction. At the early stage of modeling, an analyst may choose to pop up these summarized columns firstly for idea structuring. In other cases, he/she may do the same for objects of which the types are still unclear. For external objects, only the two fields for incoming and outgoing requests are essential.

<table>
<thead>
<tr>
<th>Services</th>
<th>Triggering Requests</th>
<th>Interface Attributes</th>
<th>Internal Attributes</th>
<th>Requests Generated</th>
<th>Incoming Requests</th>
<th>Outgoing Requests</th>
<th>All Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service1</td>
<td>Request1</td>
<td>Incoming attribute1</td>
<td>Internal attribute5</td>
<td>Request3</td>
<td>Request1</td>
<td>Request3</td>
<td>Attribute1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Incoming attribute2</td>
<td>Internal attribute6</td>
<td></td>
<td>Request2</td>
<td>Request4</td>
<td>Attribute2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returning attribute3</td>
<td></td>
<td></td>
<td>Request5</td>
<td></td>
<td>Attribute3</td>
</tr>
<tr>
<td>Service2</td>
<td>Request2</td>
<td>Incoming attribute4</td>
<td>Internal attribute6</td>
<td>Request4</td>
<td>Request5</td>
<td></td>
<td>Attribute4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internal attribute7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Attribute5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Attribute6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Attribute7</td>
</tr>
</tbody>
</table>

Table 4.1 The Object Template
The object template provides a complete yet well-organized picture for the object. The separation and connection between the object communication diagram and the object template give the modeling software a powerful technique to effectively manage the complexity of OOEM models. This structure also captures the semantics implied by OOEM.

### 4.4 Perspectives Generator

The Perspectives Generator provides another way for model construction and visualization. OOEM defines a *perspective* as "the aspects of the system necessary to describe the organizational activities invoked as a consequence of the requests of one external thing (client)" (Wand and Woo 1993). The entire OOEM model (view) is the union of perspectives for a set of given clients. Different perspectives can include different objects or different properties of the same object. A view will include all objects in the contained perspectives. Here, a client (or object) refers to a certain business role rather than a physical entity.

The Perspectives Generator utilizes the concept of perspective. It breaks the system model into perspectives. For each different perspective, there is a separate perspective diagram (PD). Figure 4.3 shows an example of perspective diagram for the business case provided at Appendix D. The corresponding object communication diagram is given by
Figure 4.4. With a central data repository, analysts can switch between the OCD and a PD anytime they want.

Figure 4.3 Perspective Diagram

As the figures depict, the perspective diagram gives the analyst some modeling and analysis advantages over the object communication diagram.

- The perspective diagram enables a *guided* way for model construction. As stated in Chapter 2, OOEM provides a rule based request propagation algorithm. By following the request propagation, the analyst can end up with an entire system model. The PD visualizes this propagation structure. It allows the analyst to
quickly follow and appreciate the propagation algorithm. In contrast to the openness of object communication diagram, the perspective diagram incorporates a systematic method of model construction.

- The perspective diagram takes a different focus than the object communication diagram. While the OCD gives the analyst a detailed look at each business unit and at the communication among different units, the PD shows how a customer request is satisfied. By emphasizing only one activity flow within the organization, the PD is more clearly structured and presented, while having fewer constructs than that of the entire OCD. It allows the user to better document customer-oriented work flows.

- The perspective diagram facilitates business process reengineering. It is understandable that processes, not organizational structures, are the objects of reengineering. Processes in a company correspond to natural business focuses, but they are often fragmented and obscured by organizational structures (Hammer and Champy 1993, P117-118). By using the request propagation analysis to break down departmental boundaries, the analyst can identify those dysfunctional, redundant, and non value-added processes with less effort.

The perspective generator inherits the request propagation and rule-driven nature of OOEM. It provides the analyst with additional power and flexibility for system analysis. With both the object communication generator and the perspective generator, the
modeling tool should adequately address the needs of analysts in dealing with different aspects of their modeling tasks.

![Diagram of an object communication model](image)

**Figure 4.4 Object Communication Diagram for Figure 4.3**

### 4.5 Subsystems Generator

A subsystem is a system whose components and structure are subsets of the components and structure of another system (Wand 1996). The modeling tool provides the capability of retrieving subsystems from the system through its Subsystem Generator.
Using the object communication diagram, the analyst can quickly generate a subsystem by performing two operations: **object exteriorization** and **request elimination**. The combination of these two operations and the visualization of subsystems not only facilitate some analysis and design tasks, but, by 'manipulating' the model, they also help redesigning the system to meet business changes quickly.

### 4.5.1 Object Exteriorization

Object Exteriorization presumably changes an internal object to an external object. It performs the following operations on the selected object:

1. Changes the object type from internal to external.

2. Removes any requests between this object and any other external objects, since there is no need to model communication between two ‘external’ objects. (The Scope Identification Rule)

3. Removes the external objects that only have communication with the selected object.

Figure 4.5 depicts a subsystem with the exteriorization of *Office Manager* object from Figure 4.4. It simplifies the diagram by only showing the components currently being
focused upon. The Object Exteriorization answers systems analysis and design questions such as:

- I am a division manager, what does the system look like from my point of view?
- Due to resource constraints, we can only automate some components now. But later on, we are going to automate the whole system. Can you give us a picture showing only the priority components now, but looking ahead to the future?
- If we outsource this business unit, what can we gain?

![Diagram](image)

**Figure 4.5 Subsystem with No Office Manager from Figure 4.4**
4.5.2 Request Elimination

Request Elimination presumably removes either an external or internal request from the system. It causes the following changes:

1. Removes the selected request from the system.
2. Following the request propagation, removes those services, attributes and spawned requests that solely exist for the selected request.
3. Removes those objects that have no further communication with the system.

Figure 4.6 shows a subsystem after the elimination of the Request for Contract Review request from Figure 4.4. In general, the Request Elimination helps the analyst to deal with the following sample management questions:

- What can we gain if we get rid of this non value-added process?
- Due to resource constraints, we can only automate some processes now. But later on, we are going to automate the whole system. Can you give us an object diagram with only priority processes now, but looking ahead to the future?
- What does the system look like if we do not meet this customer requirement?
4.6 **Semantics Reporter**

Semantics captures the dynamic behaviors of the real problem domain. A semantically consistent model adheres to the ontological principles, wins the confidence of the user, and has mapping integrity in relation to the problem domain. On the other hand, a semantically inconsistent model is either incomplete or incorrect.

Semantics can also provide modeling guidance. For example, a request changes the state of the recipient, which implies that at least one interface attribute is modified by a
request. Therefore, for each request created, the analyst also needs to find the attribute modified by that request.

A semantic engine is built into the modeling tool to empower the ontological rules behind the OOEM. Some rules are enforced. For example, the Attribute Ownership rule states that an attribute belongs and only belongs to one custodian object. This rule is ensured by model construction. Within the tool, an attribute is specified using the object template. Since every object has its own template, creating an attribute with no owner or more than one owner is impossible.

There are some other rules that the modeling tool does not enforce adherence to. Instead, diagnostic reports that warn of inconsistencies can be produced upon user request. This non-compulsory approach has several reasons. Firstly, during the early stages of model construction, a system with too many rules can only distract from the structuring and restructuring of ideas. Secondly, missing information is not uncommon in systems modeling. The CASE tool should allow for such incompleteness. Finally, a semantically incorrect model may help to identify business problems. There is no reason to prevent such a model from being temporarily produced.

The next chapter will detail the mechanisms for checking the model against ontological rules, and for recovering from model inconsistency.
Thanks to OOEM methodology, for the first time a modeling tool has the capability of performing semantic integrity checking. The built-in Semantics Reporter is definitely one of the most valuable components of the CASE tool. Meanwhile, the success of this product in building semantically sound models obviously rivals other modeling software.

4.7 Document Processor

To be a successful product, the modeling tool must permanently store modeling information. The Document Processor fulfills this role by generating a data file from the data repository.

The Document Processor sits between the data repository and the output. On the one hand, it reads data from the repository, formats the data, and writes the data to a user-specified document. On the other hand, when a model is called, the processor reads from the corresponding model file and refills the data repository.

Figure 4.7 shows part of a simplified document file. The file is text based with separate sections for each repository table. Every section starts with a table name as section header, followed by field names as column headers. This structure ensures the readability of the document.

The output document basically serves three purposes.
Firstly, it stores the model so that no information is lost after an analyst closes the program. In addition, multiple versions of the same model can be saved under different files, thus helping the analyst to play with different ideas.

Secondly, it supports information sharing among various tools and platforms. Some back-end CASE tools can read the model and perform their subsequent tasks. The document is also transferable across platforms. That means we can have compatible versions of the modeling tool for different operation systems. Each reads and writes the same file.
Finally, the well-structured text file provides additional flexibility. For example, an analyst can get a list of all system objects by simply printing out the *Objects* section. Although it is not recommended, under some circumstances, an analyst may want to directly edit the text file. For example, he/she can change a request’s name without having to implement the CASE tool.
5 System Architecture

This chapter will elaborate on the development of the modeling tool from a rather technical perspective. It will cover the techniques used for module development and model visualization.

5.1 Introduction

The application is developed with Delphi 2.0 and Paradox on Windows 95/NT platform. Delphi is a development tool for Object Pascal. It provides a comprehensive set of visual and database interfaces that address the needs of this application. Paradox is a local database management tool supported by Delphi. Its simple yet efficient design meets the requirements of the Data Repository as discussed in the previous chapter. To a large extent, the implementation techniques are shaped by the constructs and facilities available in Delphi, Paradox and Microsoft Windows.

The following sections will discuss the system architecture of the application. Firstly, a metamodel of OOEM is introduced. Then, the design and techniques for the data repository are presented. After that are the main algorithms for visualization and semantic checking. Finally, the designs for graphics and user interface are explained.
5.2 **OOEM Metamodel**

An OOEM metamodel was initially proposed by Tan (1997, p25). It captures the OOEM constructs and their relationships based upon the seven rules of the methodology. This conceptual model can be further transformed into system-oriented design models. An updated version of this metamodel is shown in Figure 5.1.

![Figure 5.1 OOEM Metamodel](image)

The metamodel depicts the constructs as boxes and the relationships as arrows. The accompanying numbers define the cardinality for each relationship. For example, 1+ means there are one or more participants in the relationship. Similarly, 0,1 indicates that there is either one or no participant.
Every relationship represents one OOEM rule or part of a rule. For instance, the service inclusion rule states that a service will be included if and only if it is invoked by at least one request. In the metamodel, this is primarily enforced by the relationship *Invoke*. That relationship, together with its cardinal numbers, states that a request invokes one service (or none for request sent to an external object), and a service is invoked by one or more requests.

Aggregation and decomposition are realized through two relationships: *Is Component of* and *Is a Composite*. The first relationship indicates which object is the composite of its component objects. The second relationship marks a request sent to a component with the corresponding request sent to the composite.

### 5.3 Database Architecture

As stated in the previous chapter, the data repository is required to store models, diagrams and system information. Model information is the basic and most important stored element. Diagrams visualize a model. These graphic representations are associated with model constructs. System information mainly describes subsystems and perspectives, which are actually model components selected according to different criteria.

Consequently, the database design should start with model representation. The metamodel already provides us with an OOEM model representation at the conceptual
level. An Entity Relationship (ER) diagram (Chen 1976) depicts the same model in a computer executable language, while incorporating more information as required by system design and implementation. It can be further translated into a database design. Table 5.1 shows the transformation relationship among these three modeling techniques (Yair 1996, p283).

<table>
<thead>
<tr>
<th>Activity</th>
<th>Model</th>
<th>Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>OOEM principles</td>
<td>Human-oriented</td>
</tr>
<tr>
<td>Design</td>
<td>OOEM metamodel</td>
<td>System-oriented</td>
</tr>
<tr>
<td>Implementation</td>
<td>OOEM ER model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OOEM database</td>
<td>Machine-oriented</td>
</tr>
</tbody>
</table>

Table 5.1 OOEM Transformation Relationship

Figure 5.2 is the ER diagram for the data repository. While originating from the metamodel, we add new entities and relationships to accommodate visualization and system information.

In the diagram, an optional participation is shown by drawing a small circle (o) on the side of the optional entity. For example, the Receives relationship indicates that an object may receive none to many requests, and a request must be sent to one object.

The diagram only shows some of the attributes that are of importance for either visualizing or understanding the corresponding entities. Among them, Generate Lines represents a set of attributes that store the topological positions of request lines on the
object communication diagram. Object type indicates whether an object is external or internal, and Attribute type distinguishes interface attributes from internal ones.

Figure 5.2 ER Diagram for OOEM CASE Tool Design
5.4 Algorithms

This section introduces the main algorithms for the implementation of the modeling tool. They are the algorithms for the visualization of different diagrams, for semantic rule checking, for decomposition and aggregation, and for the maintenance of database referential integrity.

5.4.1 Algorithm for Object Communication View

The object communication diagram shows the OOEM model objects and their relationships from an integrated viewpoint. It also provides a set of operations for insertion, modification and deletion of model constructs.

The algorithm for generating the diagram is given by Figure 5.3. It is written in pseudo-code, composed of two parts. The first part creates all the objects, and the second part creates the requests among them. The code is simplified for wider readership.
Select all objects from the data repository;
For each object selected do
Begin
  Get the object position and size from the data repository;
  Get object services from the data repository;
  Get object attributes from the data repository;
  Create the object on the diagram given the object position and size;
End;

Select all requests generated from the data repository;
For each request selected do
Begin
  Identify the source object;
  Identify the destination object;
  Get the request line positions from the data repository;
  Create the lines on the diagram that link the source object to the destination object;
  Create a text box that shows the request name.
End;

Figure 5.3 Algorithm for Object Communication View

The above algorithm is called when the diagram is initialized. After that, the analyst can work directly on the diagram through operations such as adding a new object, moving the position of an existing object, or changing the name of an existing request, etc. These changes will be saved in the data repository before a diagram is closed.

5.4.2 Algorithm for Perspective View

The perspective diagram shows the request propagation. It enables the analyst to follow the well-structured request propagation approach for model construction (Zhao 1995, p36). Figure 5.4 shows the part of the algorithm that initializes a perspective diagram. It is written in a deep-first recursive way suggested by the OOEM methodology.
Identify the object that starts the request propagation;
Create the object on the diagram at the prefixed position;
Get all requests generated from the object from the data repository;
For each request generated do
Begin
  Create an empty stack to store services;
  Call the recursive function with the request and service stack as parameters;
End;

The recursive function (the request, the service stack)
Begin
  Identify the service invoked by the request;
  Push the service into the service stack;
  If the service has been pushed twice into the service stack then
  Begin
    Show a warning message for dead loop;
    Return;
  End;
  Else
  Begin
    Identify the object receiving the request;
    Identify the interface attributes modified by the request;
    Identify the internal attributes accessed by the invoked service;
    Create the destination object on the diagram at the prefixed position;
    Create the request on the diagram using the prefixed lines;
    Show the service invoked inside the object;
    Identify the requests that spawned by the invoked service;
    For each request spawned, call the recursive function with that new request and new service stack;
  End;
End;

Figure 5.4 Algorithm for Perspective View

In the algorithm, service stacks are used to identify dead loops. A dead loop occurs if a service is called twice within any propagation route. Semantically, this means a request finally flows back to the service that sends this request out. If that were the case, the same request loop would be ever repeated in the diagram. A service stack traces all the parent services that indirectly spawn the current request, therefore preventing the dead loop from happening.
The perspective diagram uses prefixed increments for each graphic component (node and line), as illustrated by Figure 4.3. The whole diagram is refreshed whenever a user adds or deletes a propagation node.

This recursive algorithm is simple yet clear. It maps the rule-based request propagation approach into a computerized language. However, as other recursive programs, it takes significant system resources and processing time. As a result, it reduces the entire performance of the product. A non-recursive algorithm is therefore expected to speed up the program. This non-recursive algorithm may reflect the implementation artifact of the modeling tool.

5.4.3 Algorithm for Subsystem View

A Subsystem view is an object communication diagram that only shows some of the model objects and requests. The analyst selects the objects and requests to be included in a subsystem by removing the others. Recall that there are two operations for generating a subsystem: object exteriorization and request elimination.

Figure 5.5 lists the pseudo-code for these two operations. They are used to identify and store the excluded objects and eliminated requests into the data repository. The algorithm for request elimination is recursive because it will remove the entire propagation flow triggered solely by that request.
1. Object exteriorization

Identify the object to be excluded;
Add the object into object exteriorization data repository;
For each request sent out from the object do
Begin
  Find the destination object;
  If the destination object is an external object then
  Begin
    Add the request into request elimination data repository;
  End;
End;
For each request sent to the object do
Begin
  Find the source object;
  If the source object is an external object then
  Begin
    Add the request into the request elimination data repository;
  End;
End;

2. Request Elimination

Identify the request to be eliminated;
Add the request into the request elimination data repository;
If a service is invoked by the request then
Begin
  Find the number of requests that invoke the service;
  If the service is only invoked by the eliminated request then
  Begin
    For all requests spawned by the service do
    Begin
      Recursively eliminate the request;
    End;
  End;
End;

Figure 5.5 Algorithm for Subsystem View

The visualization of a subsystem is quite simple. It uses the algorithm for object communication diagram while not showing those excluded objects and eliminated requests. Those objects that have no relationships with other objects will also be removed. The excluded objects and eliminated requests can be found in the data repository, which should have been populated by the algorithms listed in Figure 5.5.
5.4.4 Algorithm for Semantic Checking

The semantic integrity-checking algorithm is realized through queries to the data repository. The program looks for ten inconsistencies in relation to the OOEM rules. Each inconsistency is checked by one SQL query with one type of error message.

The algorithm only examines the first four rules. The Attribute Ownership Rule (Rule #5) is already enforced by the data repository design. Since Rules #6 and #7 deal with the structure aspect of the system, they are also excluded from the semantic violation report.

All types of semantic error messages are listed below, sorted by their violated rules as indicated within the ()s. The request propagation algorithm was also checked when the author came out with these types of semantic errors.

- The request sent to an internal object does not invoke any service. (Rule #1)
- The request sent to an internal object modifies no interface attribute. (Rule #1)
- The request from an external object is sent to another external object. (Rule #1)
- The request from an internal object is not spawned by any service. (Rule #1)
- The service spawns the same request more than once. (Rule #1)
- This object has no communication with any other object. (Rule #2)
- The service is not invoked by any request. (Rule #3)
- The interface attribute is not accessed by any service. (Rule #4)
• The interface attribute is not modified by any request. (Rule #4)

• The internal attribute is not accessed by any service. (Rule #4)

There is an owner object for every error message. For a problem request, it could be the source or destination object depending on the checking. For a student-university system, if a student requests for university admission without an application form, the University internal object would have an error message states that “The request: student request for admission modifies no interface attribute.” Or if a student’s request for bank service is modeled, the Student external object would have a warning message states that “The request: student request for bank service is sent to another external object named Bank.” The error message itself also indicates what the owner object is.

For a problem service or attribute, the owner object would be the object that the service or attribute belongs to.

5.4.5 Algorithm for Decomposition

Decomposition and aggregation are centralized directly in the data repository. The algorithm sets up filters to select objects and requests in the database. By adding filters, only components that belong to the current object level are accessible by other functions. The filters will change dynamically when the analyst moves to another decomposition layer.
The program uses two database links to control shifts across levels: *Is Component of* and *Is a Composite*. An explanation of these two relationships can be found in section 5.2 and in Figure 5.2.

The layer control is designed in such a way that it is transparent in relation to the above-mentioned algorithms for visualizations and semantic checks. It avoids the complexity of managing decompositions by individual program functions, and therefore increases the system's performance and robustness.

### 5.4.6 Algorithm for Referential Integrity Maintenance

Referential integrity is vital to the repository's data quality. It ensures that every database relationship can be referenced and that there is no 'junk' data left in the database.

The program maintains this integrity at the coding level rather than at the database level based on three reasons. Firstly, there are many optional participation relationships that can not be managed at the database level. Secondly, database referential enforcement restricts the set of data manipulations the system can perform. Thirdly, Paradox does not support reference integrity. That means another database tool must be selected in order to set up integrity checks at the database level.
The referential integrity is carried out by delete prevention and delete cascade. Delete prevention prevents an entity from being deleted if other entities are referring it. On the contrary, delete cascade deletes an entity with all its relationships.

These two techniques are exclusively selected based on the relative importance of the entity to its links. For example, an object is delete-prevented if there are requests sent to it. On the other hand, the program automatically removes a request from the request elimination repository for a subsystem when the request itself is removed from the database.

5.5 Graphics and User Interface

The program is designed consistently with the Windows standards for graphic user interface (GUI). Many scientific studies have pointed to the quality, usability, and cognitive advantages of GUI for human-computer interactions (e.g. Card and Moran 1995; Molich and Nielsen 1990). The above researchers also suggest guidelines for interface design.

The modeling tool follows these design guidelines. Consequently, a Microsoft Windows user can easily get used to the tool. An empirical study for the GUI and other features of the product is discussed in the next chapter. At this moment, we only want to introduce two important graphic components: the visualization of objects and requests.
5.5.1 Object Visualization

An object is represented by a rectangular box in the diagrams. The object can be resized by dragging any one of the four rectangle corners. It can also be moved by dragging and dropping the whole box. These designs follow the standard Windows approaches.

Incoming requests to an object are pointed to either its left or right side while the opposite side is used for outgoing requests. This method of drawing separates the object in flows from its out flows. It clearly shows the roles an object plays. The arrangement of requests also parallels the Incoming Requests and Outgoing Request columns at the object template, as depicted in Table 4.1.

5.5.2 Request Visualization

Another important visualization design is the representation of a request in the object communication diagram. The program chooses a unique algorithm that uses up to five consecutive lines. Each line can be dragged and moved to better fit in with the diagram. Figure 5.6 shows four request representations (A, B, C, D) that this approach automatically generates for typical topological structures. It also depicts the respective request representations (A', B', C', D') after user optimization. This method is based on the assumption that a request is only linked to the left or right side of an object.
Figure 5.6 Sample Request Representation and Optimization
This request representation gives the user great flexibility to shape a request by moving line segments horizontally or vertically. By means of carefully structuring, an well-organized diagram can be finalized. However, this approach needs more user interference and imposes some loss of simplicity.
6 Empirical Study

A user study is a necessary part of the research project, as the software engineer may introduce his own perception and ‘smart’ mechanism into the system, thereby preventing it from meeting real user requirements. That is why Landauer (1995) considers personal observation of one’s own system to be one of the worst ways to assess system acceptability.

The purpose of this empirical study is multifold. Firstly, we tested whether the actual tool meets the system requirements as listed in Chapter 3. Secondly, we identified system weaknesses that need to be improved to better address user needs. Finally, the empirical study also suggests solutions to system weaknesses.

6.1 Research Methodology

This section introduces the research methodology used in the empirical study. We designed the methodology based on the above stated research objectives, with certain constraints in mind.

6.1.1 Limitations
Before we introduce the research design, we need to mention two constraints regarding the research scale and scope. These two constraints shaped the study design and the research outcomes.

The study was conducted with a small number of subjects due to limited resources. No significant mathematical or statistical conclusions can be drawn from such a small sample size. However, past research has revealed that a pool of four to five subjects was adequate for identifying major issues (Egan, et al. 1990). As a result, the study was focused on qualitative observations.

Due to system complexities, not all system features could be tested, given the limited allocated study time for each subject. User study time was limited for two reasons. Firstly, subjects were selected on a volunteer basis, and their time was limited. Secondly, an overly long session would only lead to decreased user attention and therefore, would reduce the quality of the study. Consequently, the study was designed to focus mainly on the system’s modeling capability.

6.1.2 Study Design

Eight subjects participated in the study. Based on their experience with OOEM, the subjects were divided into two groups of four. Each group had two experienced members and two less experienced ones.
One group was required to start from the perspective diagram for model construction. The other group was asked to begin with the object communication diagram. The objective of this separation was to test and compare the two ways provided by the system to build a model. For the same purpose, test participants were encouraged to switch between these two modes whenever they felt necessary. A written document, (see Appendix C), was provided to make subjects aware of the possibilities.

A pilot session was conducted first with one of the subjects. The purpose of the pilot study was to test the experimental design. As there were no major changes after that, findings from the pilot study were also included in the result analysis.

A formal session comprised six consecutive steps. The time participants took to complete each step was evaluated throughout the pilot test.

1. **Step One**

   Each test participant was required to sign the Consent Form (see Appendix A) and fill up Questionnaire 1 (see Appendix B). This questionnaire was used to collect background information on the participants' familiarity with CASE Tools and object-oriented modeling. It took about five minutes.

2. **Step Two**
The general features of the system were introduced at this step, including the object communication diagram, the perspective diagram, the object template, the semantic checking, the subsystem, and graphic operations, such as resizing and moving. The purpose of this demonstration was not to introduce system details, but to cover basic functions so that participants could start to build a model. This step took about twenty minutes.

3. **Step Three**

After the system demonstration, every participant was required to model a small business case, (see Appendix D for the case and Figure 4.4 for one proposed solution), using either or both of the model construction methods. In order to ensure that everybody received exactly the same amount of information, no questions about the case were answered at this stage. The estimated time for building the model was about thirty minutes. A participant might not necessarily have completed the model at the end.

4. **Step Four**

At this step, a subject was required to run a semantic check against the model if he/she had not already done so. If there were reported inconsistencies, he/she was also asked to attempt to fix them. The purpose was to ensure that every participant tried semantic checking, as it was one of the major features being assessed. This task took about ten minutes.
5. **Step Five**

A subject was asked to fill up Questionnaire2 (see Appendix E) after he/she had modeled the provided business case. Questionnaire2 used a series of Likert Scale questions to gather essential information for analysis. It took another ten minutes.

6. **Step Six**

A short interview was finally conducted. The interview was open-ended; its purpose was to identify more system problems, to seek solutions to these problems, and to obtain overall system performance ratings from the subjects. A set of questions (see Appendix F) had been prepared. This final step took about ten minutes depending on the time subjects had left.

Participants could stop at any of the above stages. The entire sessions were video taped for later analysis and as memory aids, with the permission of test participants.

6.2 **Subjects' Background**

Questionnaire1 provided us with background information on the subjects. Among the eight study participants, seven had more than three years experience with windowed operation systems. The last subject had about one year of Windows experience. Only one participant felt uncomfortable using Windows. Again, seven people did not think they
needed assistance with computer applications. Only one subject articulated concern for using a computer to store important information.

Of the eight people, six had education in systems analysis, but had less than one year of actual work experience. The seventh had more than fourteen years of work experience in systems analysis, while the last subject had no analysis background. They had all taken courses about OOEM methodology and its seven rules before the study. Nevertheless, four subjects had more experience than the other four.

All the subjects had had experience with some other windowed modeling tools. However, only three of them stated that they were in favor of these tools rather than just paper and pen. Two others showed negative attitudes toward the use of these tools, while the other three felt neutral.

All except one participant were highly confident in applying the object-oriented methods to diagram organizational processes.

### 6.3 Analysis of Results

Since one of the major purposes of this study was to test user satisfaction with major system features, the analysis of results was grouped according to user requirements as listed in Chapter 3. During the study, users could identify key system problems and suggest improvements to system design.
6.3.1 User Friendliness

Questions 6, 8, and 9 of Questionnaire2 were designed to test the system’s user friendliness. Question 6 asked whether the graphic operations were simple or not. Question 8 asked whether system feedback was sufficient or not. Question 9 asked whether the system was consistent with Windows or not. Table 6.1 summarizes the questions.

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Using the CASE tool, I found it easy to do graphic operations, e.g. resizing, moving.</td>
<td>Agree 4, Neutral 0, Disagree 4</td>
</tr>
<tr>
<td>8.1</td>
<td>Using the CASE tool, I found it easy to know where I was and what I had to do next.</td>
<td>Agree 4, Neutral 3, Disagree 1</td>
</tr>
<tr>
<td>8.2</td>
<td>Using the CASE tool, I found I was always prompted for inappropriate activities and incorrect inputs, as well as being given information on how to correct these errors.</td>
<td>Agree 4, Neutral 3, Disagree 1</td>
</tr>
<tr>
<td>8.3</td>
<td>Using the CASE tool, I was able to know whether a command had been completed or not.</td>
<td>Agree 4, Neutral 2, Disagree 2</td>
</tr>
<tr>
<td>9</td>
<td>Using the CASE tool, I found the user interface was consistent with Microsoft Windows.</td>
<td>Agree 7, Neutral 1, Disagree 0</td>
</tr>
</tbody>
</table>

Table 6.1 Summary of Questions for User Friendliness

We can see from the responses that, in general, subjects had positive ratings of the user interface. However, some participants were not satisfied with the graphic operations provided. The post-study interviews showed that they were mainly concerned with the
request representation in the object communication diagram (OCD). Their dislikes and suggestions included the following:

- The representation lines of a request were regenerated from a user-optimized layout to a system default layout after a connected object had been moved. The system default request layout might need some improvements, which caused extra work.
- The request line was too thin to move.
- The request line should be allowed to be connected to the top or bottom side of an object.
- The request text should be separated into multiple lines if it is too long.
- The request text should adhere to the request lines.

6.3.2 Modeling Capability

Questions 2, 3, 4, 5, 7, 10, 11, and 12 were designed to test the tool’s modeling capability. Table 6.2 summarizes all these questions. Recall that four study participants were required to start from the perspective diagram. However, all of them switched to the object communication mode for model construction after they attempted the perspective method. Meanwhile, no participant of the other group tried the request propagation approach. As a result, Table 6.2 does not treat these two groups separately.
<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Using the CASE tool, I found it easy to isolate and identify <strong>Objects</strong> from the case study notes.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>2.2</td>
<td>Using the CASE tool, I found it easy to isolate and identify <strong>Requests</strong> from the case study notes.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>2.3</td>
<td>Using the CASE tool, I found it easy to isolate and identify <strong>Services</strong> from the case study notes.</td>
<td>Agree 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>2.4</td>
<td>Using the CASE tool, I found it easy to isolate and identify <strong>Attributes</strong> from the case study notes.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>3.1</td>
<td>Using the CASE tool, I found it easy to keep track of the <strong>Objects</strong> I have entered into the system.</td>
<td>Agree 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>3.2</td>
<td>Using the CASE tool, I found it easy to keep track of the <strong>Requests</strong> I have entered into the system.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 3</td>
</tr>
<tr>
<td>3.3</td>
<td>Using the CASE tool, I found it easy to keep track of the <strong>Services</strong> I have entered into the system.</td>
<td>Agree 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>3.4</td>
<td>Using the CASE tool, I found it easy to keep track of the <strong>Attributes</strong> I have entered into the system.</td>
<td>Agree 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 4</td>
</tr>
<tr>
<td>4</td>
<td>Using the CASE tool, I found it easy to change the object type from external to internal, and vice versa.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>5</td>
<td>Using the CASE tool, I found the visualization of objects and requests helpful in the structuring and restructuring of ideas.</td>
<td>Agree 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>7</td>
<td>Using the CASE tool, I was able to check the semantic integrity of the model, and that checking resulted in fewer errors.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
</tbody>
</table>

Table 6.2 Summary of Questions for Modeling Capability
<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>Using the CASE tool, I found it easy to keep track of the links between Objects and Requests.</td>
<td>Agree 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 2</td>
</tr>
<tr>
<td>10.2</td>
<td>Using the CASE tool, I found it easy to keep track of the links between Objects and Attributes.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 3</td>
</tr>
<tr>
<td>10.3</td>
<td>Using the CASE tool, I found it easy to keep track of the links between Objects and Services.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>10.4</td>
<td>Using the CASE tool, I found it easy to keep track of the links between Requests and Attributes.</td>
<td>Agree 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 4</td>
</tr>
<tr>
<td>10.5</td>
<td>Using the CASE tool, I found it easy to keep track of the links between Requests and Services.</td>
<td>Agree 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 4</td>
</tr>
<tr>
<td>10.6</td>
<td>Using the CASE tool, I found it easy to keep track of the links between Attributes and Services.</td>
<td>Agree 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 3</td>
</tr>
<tr>
<td>11</td>
<td>Using the CASE tool, I found it easy to make modifications to the model.</td>
<td>Agree 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>12</td>
<td>Using the CASE tool, I found it easy to trace a request from its origin to the end.</td>
<td>Agree 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
</tbody>
</table>

Table 6.2 Summary of Questions for Modeling Capability (Continued)

Basically, subjects found that, using the modeling tool, they felt comfortable isolating and identifying OOEM model constructs from the case that had been supplied to them. The modeling tool also made it easy for them to track objects and services that had been entered into the system. However, a few subjects indicated that tracking requests and attributes was not so straightforward. Moreover, the representation of some construct relationships seemed convoluted to certain users. For example, four subjects stated that it was difficult for them to trace the relationships between requests and interface attributes.
Almost every subject agreed that the system provided an easy means of changing object types and of making other model modifications. In addition, the visual model representation helped them to structure and restructure ideas.

Most subjects expressed their opinions of the system’s semantic checking capability. During the post-study interviews, three participants emphasized that the semantic checking made it easy to automatically find errors, was easy to use, and had never been seen in other CASE tools they had used.

No subject built the case model from the perspective diagram; they provided the following reasons why:

- The layout of object communication and of perspective was different; that difference caused confusion when switching from one to the other.
- The OCD shows an integrated model; that helps to generate ideas.
- The perspective diagram did not show attributes directly and that made the tracking attributes difficult.
- It was much easier to identify and fix model errors in the OCD than in the perspective diagrams.

The subjects’ failure to use the perspective diagram for model construction may be the result of internal system design incapability. However, two assumptions must be verified before we jump to this conclusion. Assumption one states that unlike OCD, which
presents similarities in diagrams to other tools, the request propagation diagram is new to users; hence, they need time to get used to it. The second assumption is that a perspective diagram isolates part of the system and, therefore, makes the modeling clearer and more focused. However, the case is too simple to necessarily separate any system components. This assumption was supported by at least two study participants. Another one added that the perspective diagram was better organized and it freed him/her from carrying out graphic operations in OCD.

6.3.3 Analysis Capability

Question 13 addressed the analysis capability of the system. Table 6.3 summarizes the results.

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>Using the CASE tool, I found the visualization of request propagation helpful in the understanding of business processes.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>13.2</td>
<td>Using the CASE tool, I found the visualization of request propagation helpful in identifying those procedures that could be improved for better performance.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>13.3</td>
<td>Using the CASE tool, I found the subsystem feature helpful in isolating one part of the system from others.</td>
<td>Agree 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
</tbody>
</table>

Table 6.3 Summary of Questions for Analysis Capability

The analysis power of the system was generally affirmed by the users. Due to time constraints, most of the subjects had no chance of evaluating the business processes
provided by the case. That was why some of them remained neutral in response to this question. However, during the system introduction and post-study interview, more than half of the participants expressed the opinions that:

- The perspective diagram, which showed the process flows, visually aided the model analysis.
- The perspective diagram could trace any request back to its originator; that helped in the analysis of domain activities.
- The subsystem could assist business process reengineering (BPR) because it could provide various scenarios.
- The subsystem could be very useful in the analysis of business processes.

6.3.4 The System in General

The remainder of the questions on Questionnaire2 asked for a more comprehensive rating of the system’s performance. Table 6.4 presents the answers to these questions.
<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Using the CASE tool, I found it generally easy to build a model.</td>
<td>Agree 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 1</td>
</tr>
<tr>
<td>14</td>
<td>Using the CASE tool, it was possible to represent the important facts of the domain.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>15</td>
<td>Using the CASE tool, I will be able to build models of greater size and complexity than I would with just paper and pen.</td>
<td>Agree 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>16</td>
<td>Using the CASE tool, I will be able to build models more quickly than I would with just paper and pen.</td>
<td>Agree 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 2</td>
</tr>
<tr>
<td>17</td>
<td>Using the CASE tool, I will be able to build models that are more correct than those I could build with just paper and pen.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>18</td>
<td>Using the CASE tool, I feel confident in applying the object-oriented method to model a business domain.</td>
<td>Agree 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
<tr>
<td>19</td>
<td>Using the CASE tool, it was fun to build a model.</td>
<td>Agree 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neutral 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disagree 0</td>
</tr>
</tbody>
</table>

**Table 6.4 Summary of Questions for the System in General**

Questions 14 and 17 tested the additional effectiveness the system brought to the OOEM methodology while questions 1, 15, and 16 tested the added efficiencies. Satisfactory results were reached in terms of the answers to these questions, especially in the effectiveness area. One subject clearly indicated that semantic checking was something that could not be done by just using paper and pen.

Recall that the objective of this system is to increase effectiveness and efficiency when applying OOEM methodology. The user feedback collected from this set of questions
showed that this goal was successfully achieved. Besides, it also increased user confidence; subjects found it fun to build a model.

6.4 Suggested Research for Improvements

The empirical study discovered some system weaknesses that needed to be addressed. The subjects also suggested a number of potential improvements that could be investigated in order to turn the CASE tool to a better stage. The following list summarizes these weaknesses and suggestions:

- A better design for request representation at the object communication diagram is required. The current design is unnecessarily complicated and is inconvenient to use. The new design should consider the possibility of getting a request connected to the top or bottom side of an object. It should also allow the user to break a long request description into multiple lines.

- The perspective diagram should provide users an option to show attributes that are modified by incoming requests or accessed by request triggered services. This option would help users to trace attributes and associate them with relevant requests and services.

- A research should be conducted to explore why the perspective diagram was not used for model construction. In particular, two assumptions that may explain this should be verified. These two assumptions were stated at Section 6.3.2.
• Currently, the subsystem function is only used to display a subset of model constructs and their communications. Some users expected that the subsystem could also be used for model construction because it would allow them to concentrate on the modeling of certain business processes or units.
7 Conclusion and Future Research

This thesis continued working on the development of Object-Oriented Enterprise Modeling (OOEM) by providing a CASE tool that intended for increasing the effectiveness and efficiency of the underlying methodology. Effectiveness was realized by computerizing the OOEM rules, by computerizing the semantic integrity checking, and by computerizing the request propagation algorithm. Efficiency was realized by providing visual diagrams that facilitate model construction, model representation, and model analysis.

7.1 Contributions

This research work began in response to an urgent request to solve critical problems raised in the development of OOEMCT: model visualization, object type determination, and semantic checking incorporation. A new modeling tool was developed to solve these problems. The feasibility of the solutions has been both theoretically and empirically proved.

The capability of the new modeling tool was also extended by exploring and adding more features, made possible by OOEM. The perspective diagram and the subsystem diagram provide analysts with the ability to carry out business process evaluation and reengineering.
An empirical study was successfully conducted to test the benefits the new modeling tool can bring to the analyst. The empirical study suggested that the intended system accomplished its objectives. The study also helped to identify system weaknesses that should be improved.

This research work affirms that it is indeed important and possible to have a CASE tool for OOEM that helps the analyst to apply the underlying methodology more effectively, more efficiently, more confidently, and more pleasurably.

7.2 Future Research

The overall objectives of this thesis have been achieved. Still, much has to been done in order to turn the system into a fully functional or commercialized product. Subsequent research may focus on the following areas as suggested by the research and the user study:

- Due to the scale and scope limitations of this user study, a more comprehensive empirical study is suggested to test the new modeling tool in regard to its usefulness and user friendliness.

- A new method of request representation should be explored to overcome the identified inconveniences as stated at the previous chapter. Moreover, attributes should be displayed under their custodian objects in the perspective diagram.
- A non-recursive algorithm for request propagation is expected to reduce the system resources and response time. Theoretically, it has been proved that every recursive algorithm has at least one corresponding non-recursive algorithm.

- Both OOEMCT and the new modeling tool do not have the Sub-Classification Rule (Rule #7) implemented. A method to incorporate this rule into the diagrams should be identified. One structural approach suggested by the author is depicted in Figure 5.1 for objects and their specializations.

- This research calls for a commercialized CASE tool for OOEM. By improving the current modeling tool and adding more features such as zoom, undo and help, the finalized OOEM CASE tool will definitely benefit object-oriented modeling work.
Bibliography


Appendix A: Consent Form
OOEM CASE Tool Study
Fall 1997

Thank you for taking time off to participate in this study.

We are seeking your help in the evaluation of an Object-Oriented Enterprise Modeling Computer-Aided Software Engineering Tool developed at the Faculty of Commerce and Business Administration. The modeling tool, like other Computer-Aided Software Engineering (CASE) tools, is aimed at helping analysts build models of the domain under study so they may better understand and hence refine or resolve problems that may exist.

Your contribution and feedback will form the basis of the evaluation. As well, video taping may be done during the study or parts of it. This will facilitate the collection of important information so we may later review it. The video will NOT be used for other purposes without your prior consent in writing. The results of the evaluation may play an important part in subsequent changes or improvements to the system.

Please keep in mind that this study is to assess the system and NEVER you. In fact, reporting of the results of this study will be aggregated, and your responses CANNOT be traced back to you or any other participant. So we would appreciate your frank opinion.

Once the session is over, you are requested to NOT discuss the study with your friends and classmates.

The estimated time for this study is one hour and a half. Participation in this study is completely voluntary, and you may discontinue your participation at any time during the session.

Thank you for your participation.

Participant Acknowledgement
I have read and understood the contents of above terms and conditions and agree to participate in the study.

Name (Please Print)       Signature        Date
Appendix B: Questionnaire #1
Subject Background Questionnaire
OOEM CASE Tool Study
Fall 1997

General:

Name: (Please Print) ____________________________________________

Computer Experience Questions:

Please circle your choice where appropriate.

1. Have you ever used a windowing system e.g. Apple MacIntosh, Microsoft Windows?

   Yes / No (if No, go directly to the next question)

   • If yes, which one(s) and for how many years?
     (please circle and write length of use)
     MS Windows 3.x ________ yrs       MS Windows 95/NT ________ yrs
     Apple MacIntosh ________ yrs       X-Windows ________ yrs
     Other ___________________________/ ____________________ yrs

2. How comfortable are you using a windowing system?

   Very Uncomfortable 1 2 3 Neutral 4 5 Very Comfortable 6 7

Personal Assessment Questions:

3. I feel I need an experienced person nearby to assist me when I use the computer.

   Strongly Disagree 1 2 3 Neutral 4 5 Strongly Agree 6 7

4. I feel confident about using the computer to store important information.

   Strongly Disagree 1 2 3 Neutral 4 5 Strongly Agree 6 7
Case Tool and Modeling Experience Questions:

5. Have you taken any courses or training in Systems Analysis or related areas?
   Yes / No (if No, go directly to the next question)
   • If yes, at what level?
     Undergraduate/Graduate/Professional/Other ________________________________
   • How long was the course/training?
     __________________ months / Part-time or Full-time?
   • What CASE/modeling tool(s) were you taught in the course/training?
     (Please circle and/or write name of tool)
     SilverRun DFD OOA Tool
     Other ________________________________

6. Have you ever worked as a Systems Analyst or in a similar role?
   Yes / No (if No, go directly to the next question)
   • If yes, for how many years? __________________ yrs

7. Have you ever been taught to use the object-oriented enterprise modeling method of
   Drs. Wand and Woo (with 7 ontology-based rules)?
   Yes / No

8. From the list of CASE Tools below, indicate your level of proficiency against any
   CASE Tool you know below.
   Please circle '1' for novice user; '2' for mid-level user; '3' for advanced user.
   BridgePoint { 1 / 2 / 3 } Designer 2000 { 1 / 2 / 3 } Iconix { 1 / 2 / 3 }
   MetaEdit { 1 / 2 / 3 } ObjectMaker { 1 / 2 / 3 } OMT Select { 1 / 2 / 3 }
   OOA Tool { 1 / 2 / 3 } Paradigm { 1 / 2 / 3 } Playground { 1 / 2 / 3 }
   Rational Rose { 1 / 2 / 3 } SilverRun { 1 / 2 / 3 } WithClass { 1 / 2 / 3 }
   Other (specify) ________________________________ { 1 / 2 / 3 }
9. Having used a CASE Tool before, I would rather use a computerized CASE Tool than just pen and paper to build a model.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. What is the level of confidence you have in applying the object-oriented methods to diagram an organizational process?

<table>
<thead>
<tr>
<th>Low Confidence</th>
<th>Neutral</th>
<th>High confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix C: Two Mode of Model Building

During the construction of a model, you can always switch backward and forward between one mode and the other mode.

1. Request Propagation Mode
   General Idea:
   The entire system is triggered by request(s) from external object(s). These requests are sent to internal objects. Each request invokes a service of the destination object. The service invoked access attributes and may, in turn, spawn other request(s) to either internal or external object(s). By tracing this request propagation process in the system, analysts can identify all objects and associated services and attributes.

   Construction Steps:
   1. Identify external objects to the system and put them onto the Integrity Diagram.
   2. Pick up an external object and create a perspective sheet for that object.
   3. Switch to the perspective sheet.
   4. Do the request propagation for that object.
   5. Go back to Step 2 until no more objects are left.

2. Free Mode
   General Idea:
   Analysts identify the objects, requests, services and attributes of objects in no predefined sequence. The modeling procedure is unstructured and flexible.

   Construction Steps:
   1. Identify objects.
   2. Identify the requests among objects.
   3. Identify the services invoked by requests.
   4. Identify the attributes accessed by services and requests.
   5. Repeat Step 1 to 4 in any orders until all of them are identified.
Appendix D: Case Study
OOEM CASE Tool Study
Fall 1997

Below is a description of current activities in the Engineering Firm call Sue-Bee Associates.

Customers solicit bids on contracts, and grant or deny Sue-Bee’s bid by comparing it with other bids received. Sharon, the office manager, coordinates this activity. The client will send Sharon the project information, which includes drawings of the proposed job. She then turns these drawings over to the architects, who make detailed drawings of the job, along with cost estimates.

The project manager, Stan, uses the drawings and the estimates to develop the bid for the contract. In order to accomplish this, he also reviews the suppliers and sub-contractor’s files to determine material and labor costs. In addition to the bid, a contract is also prepared. The contract is reviewed by the firm’s lawyers, and is then mailed to the client along with the bid. Copies of the bid and the contract are sent to Sharon who then stores them in the "Main Contract" file.

If either the client or the firm ends negotiations, the bid and the contract are marked void in the "Main Contract" file.

If the contract is granted to Sue-Bee, then a summary card is begun for that project. Throughout the life of the job, Sharon will be updating the summary card for each contract with information pertaining to time spent and materials used.

At the end of each month, Sharon is required by the head office to produce an item summary report, which summarizes all of the items on the summary card for that project. Based on this report, an invoice is prepared for the client.
Appendix E: Questionnaire #2
Study Feedback Questionnaire
OOEM CASE Tool Study
Fall 1997

Name: (Please Print) __________________________________________

Please circle your choice where appropriate. You are welcome to add remarks or explain the reasons for your choices.

1. Using the CASE Tool, I found it generally easy to build a model.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. This question is broken down into 4 related sub-questions. They differ only in the **bold** text.

   • Using the CASE Tool, I found it easy to isolate and identify **Objects** from the case study notes.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   • Using the CASE Tool, I found it easy to isolate and identify **Requests** from the case study notes.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   • Using the CASE Tool, I found it easy to isolate and identify **Services** from the case study notes.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   • Using the CASE Tool, I found it easy to isolate and identify **Attributes** from the case study notes.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. This question is broken down into 4 related sub-questions. They differ only in the **bold** text.

- Using the CASE Tool, I found it easy to keep track of (remember) the **Objects** I had entered into the system.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7

- Using the CASE Tool, I found it easy to keep track of (remember) the **Requests** I had entered into the system.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7

- Using the CASE Tool, I found it easy to keep track of (remember) the **Services** I had entered into the system.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7

- Using the CASE Tool, I found it easy to keep track of (remember) the **Attributes** I had entered into the system.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7

4. Using the CASE Tool, I found it easy to change the object type from external to internal, and vice versa.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7

5. Using the CASE Tool, I found the visualization of objects and requests helpful in the structuring and restructuring of ideas.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7

6. Using the CASE Tool, I found it easy to do graphic operations, e.g. resizing, moving.

  Strongly Disagree 1 2 3 Neutral 4 5 6 Strongly Agree 7
7. Using the CASE Tool, I was able to check the semantic integrity of the model; this checking resulted in fewer errors.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8. This question is broken down into 3 related sub-questions in regard to the feedback from the CASE Tool.

- Using the CASE Tool, I found it easy to know where I was and what I had to do next.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Using the CASE Tool, I found I was always prompted for inappropriate activities and incorrect inputs, as well as being given information on how to correct them.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Using the CASE Tool, I was able to know whether a command had been completed or not.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. Using the CASE Tool, I found the user interface was consistent with Microsoft Windows.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10. Using the CASE Tool, I found it easy to keep track of the links (relationship) between the following in the system ... (Please circle inside the table)

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Requests</th>
<th>Attributes</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>1 2 3 4 5 6 7</td>
<td>1 2 3 4 5 6 7</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>Requests</td>
<td></td>
<td>1 2 3 4 5 6 7</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td></td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>Services</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11. Using the CASE Tool, I found it easy to make modifications to the model.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12. Using the CASE Tool, I found it easy to trace a request from its origin to the end.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13. This question is broken down into 3 related sub-questions in regard to the analysis of business domain based on an OOEM model.

- Using the Case Tool, I found the visualization of request propagation helpful in achieving an understanding of business processes.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Using the CASE Tool, I found the visualization of request propagation helpful in identifying those procedures that could be improved for better performance.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Using the Case Tool, I found the sub-system feature helpful in isolating a part of the system from others.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14. Using the CASE Tool, it was possible to represent the important facts of the domain.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15. Using the Case Tool, I will be able to build models of greater size and complexity than I would with just pen and paper.

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Neutral</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
16. Using the CASE Tool, I will be able to build models more quickly than I would with just pen and paper.

   Strongly Disagree 1 2 3 4 5 6 7
   Neutral           1 2 3 4 5 6 7
   Strongly Agree    1 2 3 4 5 6 7

17. Using the CASE Tool, I will be able to build models that are more correct than those I could build with just pen and paper.

   Strongly Disagree 1 2 3 4 5 6 7
   Neutral           1 2 3 4 5 6 7
   Strongly Agree    1 2 3 4 5 6 7

18. Using the CASE Tool, I feel confident in applying the object-oriented method to model a business domain.

   Strongly Disagree 1 2 3 4 5 6 7
   Neutral           1 2 3 4 5 6 7
   Strongly Agree    1 2 3 4 5 6 7

19. Using the CASE Tool, it was fun to build the model.

   Strongly Disagree 1 2 3 4 5 6 7
   Neutral           1 2 3 4 5 6 7
   Strongly Agree    1 2 3 4 5 6 7
Appendix F: Post Study Interview Record
OOEM CASE Tool Study
Fall 1997

Name: (Please Print) ________________________________

1. How do you feel about this modeling tool compared to other CASE tools that you have used?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

2. What are the strengths of this system?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

3. What are the weaknesses of this system?

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
4. What are the specific areas that need to be improved immediately?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

5. Please include here any other comments that you want to make.

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________