MEASURING STRUCTURAL COMPLEXITIES OF MODULAR, HIERARCHICAL LARGE-SCALE MODELS

by

Sumitha Mohan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

August 2003

MEASURING STRUCTURAL COMPLEXITIES OF MODULAR, HIERARCHICAL LARGE-SCALE MODELS

by

Sumitha Mohan

has been approved

July 2003

APPROVED:

_____, Chair

_____

_____

Supervisory Committee

ACCEPTED:

_____

Department Chair

_____

Dean, Graduate College

ABSTRACT

Large-scale systems are increasingly being developed using model-based analysis and design techniques. The products of these techniques are a set of models summarizing architectural and detailed design decisions that show scalability and complexity traits. Scaleable modeling is expected to support the specification of many hundreds to thousands of interacting model components. Simple "structural complexity measures" of a model are its number of components, number of alternative component types, and the number of component-to-component interconnections. Therefore, the purpose of this research is to study a modeling methodology that (i) supports scaleable specification and (ii) facilitates measuring structural complexity metrics. Using such an approach, modelers can quantitatively evaluate complexities of their analysis findings and/or design choices.

This research extends the Scaleable Entity Structure Modeler (SESM), an existing modeling environment that is suitable for characterizing modular, hierarchical systems. This environment supports modeling persistent large-scale models by providing a user with multiple, complementary, architectural structures of a system in terms of its model components and couplings. This thesis presents the software design and implementation of an extended SESM environment called Scaleable Entity Structure Modeler with Complexity Measures (SESM/CM). This environment offers a suitable basis for representing and computing structural complexity metrics. To show the applicability and use of the SESM/CM environment, it is used in the domain of computer networks modeling. The contributions and results of this thesis are the implementation of the SESM/CM, a case study for the ASU Campus Computing Network (ASU-CCN), a model developed for a part of the ASU-CCN, and a discussion on its structural complexity measures.

ACKNOWLEDGMENTS

I wish to offer my sincere gratitude to my advisor Professor Hessam S. Sarjoughian, for his continued advice, patient support and encouragement that made the completion of this work possible. I would like to thank Professors James Collofello and Susan Urban for serving on my thesis committee.

This research is supported under the NSF Scaleable Enterprise System Grant No. DMI-0122227. I gratefully acknowledge NSF's financial support toward completing my studies.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER 1 INTRODUCTION 1

1.1. Approach 3

1.2. Contributions 5

1.3. Overview of the Thesis 6

CHAPTER 2 BACKGROUND 8

2.1. Scaleable System Entity Structure Modeler 8

2.2. SESM Architecture 9

2.2.1. User Interface 11

2.3. Comparison of SESM with other Modeling Tools 18

CHAPTER 3 ASU NETWORK CASE STUDY 20

3.1. The 3-tier Model 20

3.1.1. The Core Layer 20

3.1.2. The Distribution Layer 21

3.1.3. The Access Layer 22

3.2. ASU Campus Computing Network Design 22

3.2.1. Backbone Topology 22

3.2.2. Basic Building Block/Module 25

3.2.3. Main Components of the ASU Computing Network 25

Page

CHAPTER 4 SYSTEM MODELING COMPLEXITY MEASURES 30

4.1. Concepts for measuring model complexity 30

4.2. Metrics Table 33

4.3. SESM Database 35

4.3.1. Entity Relationship Diagram 35

4.4. UML Data Model Diagram 39

CHAPTER 5 DESIGN AND SPECIFICATION OF SESM 48

5.1. Design Requirements 48

5.2. Extensions to the Relational Database 50

5.3. Extensions made to SESM Server 51

5.3.1. Add Operations (Add / Modify Transactions) 53

5.3.2. Delete Operations (Delete / Modify Transactions) 57

5.3.3. Extended and Newly Added Methods: 60

5.4. Extensions made to SESM Client 64

5.4.1. User Interface Enhancements in SESM Client 66

CHAPTER 6 RESULTS, FUTURE WORK AND CONCLUSIONS 69

6.1. ASU Campus Computing Network Model in SESM 69

6.2. Future Work 76

6.2.1. Computation of Additional Metrics 76

6.2.2. Support for Collaborative Model Development 76

6.2.3. Support for storing Multiple Models 77

6.3. Conclusions 77

Page

REFERENCES 80

LIST OF TABLES

Table Page

1. Network Devices and their Description 27

2. Network Links and their Description 27

3. ER Diagram Explanation 37

4. Required Table Constraints and their Cardinality 43

5. Relational Database Schema Specification for Metrics Table 51

LIST OF FIGURES

Figure		Page
1.	SESM Client-Server Architecture	9
2.	SESM System Components Overview Diagram	11
3.	SESM Graphical User Interface	12
4.	Enhanced SESM Toolbar	14
5.	Component Selection List	15
6.	Number of Components Option	15
7.	ModelA With Components and Coupling	16
8.	A separate Model Template to simulate dynamics of Cable	17
9.	Coupling with Name Assigned	18
10.	The 3-tier Model	21
11.	Building Layout Along with Router Placement	23
12.	Core Layer Topology	24
13.	The Building Block	25
14.	ASU Campus Computing Network Diagram	28
15.	Sample of Metrics Associated with a COUPLED Model in SESM/CM	35
16.	SESM ER Diagram	36
17.	An Identifying Relationship Example	42
18.	An Intersection Table Example	44
19.	Relationship between Metrics table and ModelTemplate table	44
20.	Data Model Diagram of SESM/CM Database - View 1	46
21.	Data Model Diagram of SESM/CM Database - View 2	47
22.	SESM Server Use Case Diagram	49

Figure Page

23. SESM/CM Architecture - Parts Modified 52

24. Metrics Use Case Diagram 54

25. SESM Server Component Digram 61

26. Algorithm for Add Component Operation 63

27. Add Component Operation on SESM Server Sequence Diagram 65

28. Metrics Consistency and Uniformity 66

29. SESM Client Sequence Diagram 67

30. GUI representing Complexity Metrics for COUPLED Model 68

31. GUI representing Complexity Metrics for ATOMIC Model 68

32. Large-Scale Model of the ASU Campus Computing Network in SESM/CM 70

33. University Network SubModel 71

34. Zone1 SubModel 71

35. GWC Network SubModel 72

36. Floor 3 of the GWC Network 72

37. Metrics for Zone1 73

38. Metrics for ACIMS Lab 74

39. Instance Template Model View of University Network 75

40. A Possible Extension to the Couplings Metric 76

CHAPTER 1

INTRODUCTION

Since the advent of computing, engineered systems have continued a spiraling trend toward greater degree of sophistication that translates to increased functionality and complexity. Industries such as telecommunication, healthcare, government, educational, and entertainment depend solely on computer-based systems. These systems are widely being used to control and manage the natural resources, health-care delivery, and economies of the world. Indeed commerce, medicine, law, and politics among others have been transformed greatly during the 20th century to a large part due to the information technology revolution.

These computerized systems provide the basis for today's world to operate in a seamless fashion along multitudes of organizations and societies. For example, many universities across the globe, knowingly or not, are fashioning a "universal education system" by making possible sharing of research findings, course lectures, publications, etc. based on the wide variety of integrated services offered through the Internet. To be sure, the future of the world will be stark and unimaginable should we find ourselves without the computing resources we rely on for our day to day livelihoods. Henceforth, given the complexity and scale of existing systems, and others even with greater complexity on the horizon, it is imperative to understand their structural and behavioral intricacies. Furthermore, inter-dependencies

among these systems is inevitable as our world continues toward total integration across cultural, economic, and national boundaries. Consequently, it is imperative to be able to systematically understand the inner working of each system and its components, and also for a system of systems.

One scientific approach to represent the structure of such large, complex systems is via computer modeling. In fact, nearly all existing information-based systems have been developed using multiple types of static and dynamic model characterizations such as classical data and control flow diagrams [Pre97] and modern class and statechart diagrams [Har87, Boo98]. One important technique to modeling of large-scale systems is through capturing a system's architecture in terms of its parts and their interactions [Bass98]. This approach allows making scientific inquires about static and dynamic behavior of system's structures. For example, we can quantify and measure static aspects of a system in terms of its number of components and the number of ways they interact. Similarly, we can quantify and measure dynamic aspects of such a system in terms of its performance, reliability, and robustness. These metrics provide a basis to formulate new ways of measuring a system's complexity.

For example, given a system such as Arizona State University's (ASU) computer network, we can determine the total number of components, the number of different types of components, and the number of interconnections. With the ability to specify and determine a system's simple **structural** (static) metrics such as number of components and interconnections, we can develop a new genre of multipart (high-level) scalability, interoperability, and reusability metrics. These metrics can collectively lead to aggregate, system-wide complexity metrics which can be used toward open research inquires on model composability and therefore modeling and simulation verification, validation, and accreditation.

The overall aim of this thesis, therefore, is to develop and implement a subset of the above concepts and methods, and approach to measuring structural system complexity. In particular, in this thesis, we will extend the Scaleable Entity Structure Modeler [Sar03, FU02] environment to capture and compute persistent metrics such as the number of components and number of interconnecting links for modular, hierarchical system-theoretic models.

1.1. Approach

To develop the above concepts and design a set of metrics for modular, hierarchical systems, we have chosen modeling a computer network system. This kind of system is ideal since its structure can be modeled hierarchically and typically can have many thousands of components and interconnections. Additionally, these systems are increasingly being developed using simulation-based design and therefore having their models can play a key role in the architecture specification and detailed design. Hence, analysts and designers interested in the complexity measures of such systems will be able to make well-defined quality of their design decisions to support required performance, robustness, and other quality attributes of interest.

There exists numerous general-purpose and specialized modeling environments such as Matlab [Mat98], OpNet [Opn], Rapide [Bry98], and Rational Rose [Ros00]. These modeling tools are often used to create models at various levels of abstractions including structural decomposition and state-based dynamic behavior. These tools support modeling systems using the concept of a structure composed of nodes that are interconnected with arcs. Unfortunately, to our knowledge, no existing modeling environment, other than the Scaleable Entity Structure Modeler [Sar03, FU02] has been conceived to support alternative charac-

terization of a system's structures having many thousands of components (nodes) and links (arcs).

Scaleable Entity Structure Modeler is a modeling tool for characterizing modular, hierarchical systems. The underlying modeling methodology of SESM supports growth in model size and complexity. However, this capability is not supported in other existing modeling approaches (e.g., Modelica [Mod00] and UML [UML00]) having an underlying rigorous methodology. It provides a modeling engine with a graphical user interface for creating, modifying, storing, and reusing atomic and composite model components. The primitive modeling constructs it supports are decomposition (part-of) and specialization (is-a) relationships among model components. Its database allows storage of models using a relational database management system and therefore it is suitable for the development of large-scale models.

Automated tools (both general and specific) for specification of models has long been used to to analyze the design and behavioral aspects of software or system architectures. Examples are Rational Rose [Ros00] and OPNET [Opn]. Rational Rose is a general purpose modeling tool that aids in building complex software systems. OPNET is a computer network specific tool that helps in the specification and simulation of hierarchical, complex network system models. However no modeling methodology or environment has been devised to enable measuring the kinds of structural complexities discussed above for system models. Manually determining structural complexity (e.g., number of components and number of interconnections) is impractical for even modest size models. This thesis, therefore, seeks to extend the SESM's capability to enable computing some complexity metrics of large-scale models created using the Scaleable Entity Structure Modeling approach.

This thesis introduces SESM/CM (Scaleable System Entity Structure Modeler with

Complexity Metrics) an extended version of the SESM modeling environment. SESM/CM uses the underlying SESM for creating, modifying and reusing models but extends SESM's capabilities to facilitate computing basic structural complexities of the models. Thus this research has developed and implemented computation of the basic structural complexity metrics for model design. The metrics we focus on are purely structural and it is very important to note that we donot consider any behavioural (dynamic) aspects of the system.

1.2. Contributions

Designed and implemented capabilities into the SESM to support capturing and viewing complexity measures of modular, hierarchical system models - New capabilities were introduced into the SESM's client, server, and database modules. SESM/CM, the extended version of SESM uses the best practices of object-oriented software analysis and design. In particular, the overall design of the SESM's architecture is retained while the complexity metrics computations are added at the appropriate parts of the system. Specifically, the server, client and database parts of the SESM were extended to support capturing, monitoring, and storing complexity measures.

Developed a conceptual model of the ASU's Campus Computing Network - The conceptual model focuses on the structural aspect of the network and in particular emphasizes on the scale of the network, different kinds of nodes and links in the network, and organization of sub-networks across the entire network. The ASU network was chosen to be the testing ground because we had access to the domain experts and could gather information about the ASU's network infrastructure [Rep03].

Developed a medium-scale network model of the ASU Computing Network - The case study considers and models the key elements of the ASU Campus Com-

puting network. The network topologies and their components were modeled based on a series of interviews with the Arizona State University Technical Support Team. We collected information about some of the key structural aspects of the network - i.e., three-layer architecture of the network, types of devices and links, and the overall network design. This information provides the basis for developing the ASU's computing network model in the extended SESM environment.

Demonstrated the applicability of the approach on the ASU network model - The correctness of the design and implementation were shown using the ASU's case study model.

1.3. Overview of the Thesis

The thesis is organized as follows. In Chapter 1, we have summarized the rationale and importance of the proposed research and our approach in achieving the thesis outcomes.

Chapter 2 describes the System Entity Structure Modeler tool and some aspects of its design and capabilities that serve as a foundation for introducing and supporting complexity measures.

Chapter 3 presents a case study aimed at modeling the ASU Campus Computing Network (ASU-CCN) and hence the development of the ASU-CCN model in SESM. In this chapter, we exemplify the role of complexity measures and what benefits it can provide to ASU's analyst, designers, and administrators.

Chapter 4 discusses the key concepts and methods for defining and measuring complexity measures of hierarchical systems based on the SESM modeling approach. In this chapter, we will introduce a generic model for capturing metrics of interest. An Entity-Relation representation of the metrics and its UML specification are described.

Chapter 5 describes detailed software design and implementation of the SESM/CM - an extension of the SESM environment capable of capturing and computing *Complexity Measures*. In particular, this chapter focuses on detailed complexity metrics computation capability in terms of client, server, and database modules of the SESM/CM. This chapter also discusses the details of the graphical user interface of the metrics computation.

Chapter 6 presents a model of the ASU Campus Computing Network (ASU-CCN) developed using SESM/CM. This model illustrates the applicability of the complexity metrics as applied to a small part of the ASU-CCN model. Finally, this chapter suggests some additional future work and summarizes the contributions of this research.

CHAPTER 2

BACKGROUND

2.1. Scaleable System Entity Structure Modeler

SESM is a modeling environment that allows users to create hierarchical "object-like" models using a relational database and a user interface [FU02]. Models can be stored and maintained by a relational database management system (RDBMS) and the graphical user interface provides a friendly menu for creating, modifying, and manipulating hierarchical modular models. Every model can be considered as an object having input and output ports. The Scaleable System Entity Structure Modeler (SESM) can represent system-theoretic atomic and hierarchical coupled models.

An *Atomic Model* in SESM has well-defined interfaces (i.e., every component has input and output ports) that are necessary for its interactions with any other model. A *Coupled Model* has components and couplings. The components of a coupled model are related to each other through *Internal Couplings*. The components are related to the coupled model through *External Input Couplings* and *External Output Couplings*. With SESM users can capture "structural" representation of hierarchical models. SESM may also be extended to support varying degrees of modeling using collaborative technologies such as the Collaborative Distributed Network System [Sar99].

Figure 1. SESM Client-Server Architecture

2.2. SESM Architecture

SESM is composed of a database management system (RDBMS), a network environment, a modeling engine (Server), and a user-interface (Client) as shown in Figure 1. The model data is stored in a relational database. The server can initialize and manipulate the database based on users requests. The client displays models for the user and enables modifying them. Both the client and the server are connected to the DBMS via a network environment. Server and client independently initialize and maintain their connectivity. This separation allows user interactions requiring "write access" to the DBMS to be mediated by the server while "read-access" interactions to be carried out directly with the database.

Scaleable System Entity Structure Modeler supports multiple users concurrently accessing and manipulating models that are stored in the database. It provides the management mechanism to keep the data consistent in addition to the facilities provided by the DBMS. SESM has a server that writes to the database, with potentially multiple clients interacting with the database. Clients are also connected to the DBMS, allowing them to retrieve model data concurrently. But the client must connect to the server in order to

modify the database. This architecture allows a single writer but multiple readers to be in the database concurrently and thus provides a restricted flavor of network environment.

The client can modify the model data by sending requests with required parameters to the server. In this architecture, the clients' requests to the server are serialized if they are intended to modify the model. The serialization ensures that the server only receives one request at any time. When the server receives a modification request, the server broadcasts a notification to all the clients. When each client receives the server notification, the client reads the model data directly from the DBMS. However, if the server did not complete the modification, the server sends out a message to the specific client that is requesting the modification with a reason of failure. Thus, the client also receives feedback of its incomplete tasks from the server.

This architecture does not generate superfluous network traffic between the clients and the server. Thus, it provides better scalability since a large number of queries have been shifted from the server to the DBMS. Retrieving the model data directly from the database has an additional benefit. It makes the development of the client and the server less dependent.

By design, the SESM system includes the SESM package, the network environment package, SESM client, and SESM server as shown in Figure 2. The SESM package should serve as an API used to access the SESM representation model data stored on the DBMS. There are three main components in the SESM package; Connectivity, SESM Query, and SESM Modifier as shown in Figure 2.

The connectivity component is used to connect to the DBMS. It handles all the communication between the SESM system and the DBMS. The SESM query component retrieves

Figure 2. SESM System Components Overview Diagram

data from the DBMS using SQL and maps the data into object-oriented SESM models. The SESM modifier component modifies the SESM representation models on the DBMS. This component translates the requested modification into appropriate SQL statements. The server extends the server provided by the network environment package. Messages received by the SESM server are processed, and modifications are performed accordingly. The SESM client utilizes the SESM query component to retrieve and display the SESM representation model visually on its graphical user interface (GUI). The user also modifies the model through the SESM client's GUI. The network environment package manages the communication between the SESM client and the SESM server by providing the components that can be extended by the SESM client and SESM server.

2.2.1. User Interface. Figure 3 shows the Graphical User Interface window of SESM. As you can see there is a Menubar that has Operations and Database Menus. The Operations menu has options for the following,

- Create Template Model

Figure 3. SESM Graphical User Interface

- Create Instance Template Model
- Create Instance Model

The Database menu has option to initialize the database. Upon initialization, the model data is totally erased. This saves the user from the task of deleting every single model template in case he needs to delete all model templates.

The Model Tree shows the tree view of the hierarchical model. It is controlled by the tabs and alternates between Template Model View (TM), Instance Template Model View (ITM) and Instance Model View (IM) depending upon the tab chosen.

The Model Block Viewer area of the GUI shows the GUI View of the model node that is currently selected in the Model Tree. It shows the model along with its ports, couplings and components. In this thesis along with the main focus of computing complexity metrics, we have also enhanced the SESM User Interface with additional features to improve user's convenience that were lacking in the previous version. Overall these are the areas that were enhanced.

- Added Functionalities to the Toolbar
- Added Functionalities to the Option Window
- Assigning Names to Couplings

These enhancements have no relation to the complexity metrics computation. The above features were added with the intention of having more user-friendly capabilities to the GUI. The following section explains each of them in brief.

2.2.1.1. *Functionalities Added to the Toolbar.* Figure 4 shows the enhanced toolbar. It has the newly added buttons for Print and Save. The Print button allows the user to

Figure 4. Enhanced SESM Toolbar

print the current user interface screen through his local/network printer. The Save button allows the user to save the GUI currently on screen (the tree view and the model block view of the selected node) as a JPEG image in his specified directory location.

2.2.1.2. *Functionalities Added to the Option Window.* When users want to add a component to an existing Model Template they can use the Add Component menu item present in the model menu. Thereby the user is presented with the list of available model templates from which he can choose the Template Model that he intends to add as a component. This works fine when the user adds a small number of components such as 3 or 4. But when the user creates large-scale models many a time he may encounter a situation where he may want to add, multiple similar components to a parent model. For example, while creating a network model of the ASU Campus Computing Network we wanted to add 20 Computers to a Research Lab. During those kind of situations it is a cumbersome process for the modeler to manually add the component 20 times. In this thesis an attempt has been made to eliminate this problem by giving an option to the user to specify the number of components whenever he intends to add such that he can add multiple components using a single command. Figures 5 and 6 show the process.

From Figures 5 and 6 it is seen that when the user chooses to add the component Computer to Research Lab1, SESM immediately asks the user how many computers needs

Figure 5. Component Selection List

Figure 6. Number of Components Option

to added. When the user specifies a finite integer 20, the parent component Research Lab1 is populated with 20 occurrences of computer.

2.2.1.3. *Assigning Names to Couplings.* In SESM, the connectivity between models is accomplished through couplings. A coupling connects a port in one model to a port in another model. Depending upon the relationship between the models (Parent-Child, Component-Component) for which coupling is established and the type of ports, the coupling may be any one of the following types.

- Internal Coupling
- External Input Coupling
- External Output Coupling

Figure 7. ModelA With Components and Coupling

In a network model couplings are very vital to represent the connection between the networked hosts. For e.g., consider a sample Model A in which a switch is connected to a router through a CAT 5 cable. Figure 7 shows the model.

There are 3 possible cases that may arise because of this modeling scenario

1. The modeler intends to simulate the dynamics of the cable
2. The modeler does not intend to model the cable dynamics nor does he want to specify any details about the cable
3. The modeler does not intend to model the cable dynamics but wishes to specify the some details about the cable, say, type of the cable (CAT 5)

In Case(1), since the cable's behavior needs to be simulated it will be essential to create a separate model of the cable. So the picture would have been as in Figure 8. Case(2) does not need to simulate the dynamics of the cable nor specify details about it. This is the simplest case where the create model looks like Figure 7.

Consider Case(3), where the modeler does not intend to model the cable dynamics but wishes to specify some details about the cable such as the type of the cable that is CAT5.

Figure 8. A separate Model Template to simulate dynamics of Cable

The modeler needs to represent that the router and the switch are connected through a CAT 5 cable. It would be a bad idea to create an individual model for CAT 5 just for the sake of representation without the need to simulate. For a small-scale system the impact on the performance will not be as compared to a large-scale system, where such unnecessary components are a high price to pay. Imagine there are a 200 occurrences of Model A in the system then we would be increasing the unwanted complexity by adding 100 CAT 5 models having 200 ports and 400 couplings. This situation is handled in this thesis by providing an option to the modeler to assign a name to the coupling. The coupling menu has been improvised to provide a function that facilitates optional assignment of name to the coupling. The coupling name is optional and may be named at any time the user wishes to, i.e., it is not restricted to the time of creation of coupling. Figure 9 shows the GUI where the coupling is named. The router and switch are connected using a cable that is named CAT 5.

Figure 9. Coupling with Name Assigned

2.3. Comparison of SESM with other Modeling Tools

As systems being built are becoming more complex in their structure the need for proper modeling tools is also increasing. The model acts as a blueprint for the construction of the system and also helps to maintain, improve and troubleshoot an existing system. Having a wide choice of tools for modeling like OPNET and UML it is imperative for the modeler to compare the capabilities each of these tools can offer in order to decide the tool that best matches with his requirements. This section compares SESM with a few other well-known modeling tools in terms of providing the complexity measure for the model developed using the corresponding tools.

UML (Unified Modeling Language) is a tool for specifying software systems. It provides project life-cycle support (with class diagrams, sequence diagrams etc.) for developers, data modeling for database designers and many other capabilities that simplify the process of software design.

OPNET (OPtimizing Network Engineering Tools) is a tool used for simulation of communications systems, protocols and networks.OPNET is capable of simulating large

communications networks with detailed protocol modeling and performance analysis. OPNET is a full-fledged hierarchical network modeling and simulation environment that provides a model library constituting of nodes and also facilitates simulation.

However the above tools do not support computing the kinds of complexities associated with large-scale system models. As in SESM the above-mentioned tools have no option for measuring the structural complexities of the system.

The metrics provides designers and modelers who may not be absolutely familiar with the design and details of the system, with an indication of the scale and complexity of the system. By using metrics they can identify parts of the system that may require more testing and also the areas that need re-design. Also potential bottlenecks in the system may be identified early in the process of design.

For instance, load-balancing is a very important factor that affects the performance of network systems. Load-balancing is the distribution of processing and communications activity evenly across a computer network so that no single device is overwhelmed. By using the complexity metrics computation option provided by SESM on major internetworking components like routers and gigabit ethernet switches, load-balancing can be effectively performed by knowing the number of network devices (components) connected to each of them. Re-designing the network depending on the information gathered from SESM's metrics to optimize load-balance the modeler can ensure better performance and increased availability.

CHAPTER 3

ASU NETWORK CASE STUDY

Campus Computer Networks usually connect thousands of networked devices and are essentially large-scale systems. To illustrate model creation using SESM the campus network of ASU was used as an example. Campus network was chosen to be the ideal system for creating a SESM sample large-scale model since it satisfies all the 3 criteria of a system that can be successfully modeled using SESM. Campus networks are large-scale, hierarchical, modular. As already mentioned they are inherently large-scale and the section below justifies the hierarchical and modular nature of campus networks.

3.1. The 3-tier Model

ASU campus network is a classical example of a hierarchical three-tier model. The campus network has a core or backbone that spawns/forks into the entire structure. The network model defines 3 levels or functions of the network - *Core, Distribution* and *Access* (See Figure 10). This model allows for scalability since it provides expansion points at each of these three layers without affecting the other portions of the network.

3.1.1. The Core Layer. The core layer usually focuses on Wide Area Network (WAN) devices connected by links that span geographically diverse locations. It also in-

Figure 10. The 3-tier Model

cludes the access point of the campus to the Internet. The usual devices present in this layer are high-speed routers and links are usually Fiber Optic. This layer is responsible for the high-speed transfer of data. It acts as a high-speed switched backbone. The core layer routers do not have services like access-lists that would reduce the efficiency of the network. This layer is designed to have redundant paths and other fault-tolerance mechanisms. The main goals of this layer are providing path redundancy, load-balancing, rapid convergence and efficient bandwidth usage [Tho01]. Core routers perform the most important function of a network - they switch packets at high speed. So these routers are connected via multiple paths or meshed (as shown in Figure 12) for providing redundancy. Servers, workstations, and other peripheral devices are not placed in the core.

3.1.2. The Distribution Layer. The distribution layer sits between the core layer and the access layer. It connects the campus backbone to all the remote sites via Layer 3 or Layer 2 switches. Also known as the workgroup layer, it provides services to the workgroups and departments. The usual design goals of the distribution layer is to provide and control access to the services on the network, implementing of network policies, addressing schemes,

defining naming conventions, path metrics and control network advertisements [Tho01]. Security is implemented in this layer by access lists which is a mechanism by which users are given access only to the services they need. distribution lists, map lists, static routes are defined to control the flow of traffic at the distribution layer. This layer concerns with terminology and technologies like Ethernet, Fast Ethernet, Gigabit Ethernet, Asynchronous Transfer Mode(ATM) and Fiber Distributed Data Interface(FDDI).

3.1.3. The Access Layer. The access layer connects users to the network and network services. This is where the user literally plugs their network device into their local switch through appropriate wiring closets or jacks. It is the point where the users first access the network and network services. Severs, workstations, print and other peripheral devices, dial-up services are implemented at the access layer. Functions like grouping of like users into a domain is done at this layer. We usually talk about terms like ethernet, hubs, modems, servers and workstations in the access layer.

3.2. ASU Campus Computing Network Design

3.2.1. Backbone Topology. In a campus having many buildings the required number of backbone devices (which may be high-end routers, or switches that have routing capability) is decided first, say N. This decision is made taking into account factors like the size of the university, the approximate number of users and estimated traffic. The next step is to decide where to place these devices. The campus is divided into N zones. These devices are placed into geographically dispersed buildings such that every device is placed in a building belonging to a different 1/Nth zone of the campus.

Figure 11. Building Layout Along with Router Placement

Figure 12. Core Layer Topology

Figure 11 shows the campus building layout of ASU and the placement of routers in different buildings. The campus is divided into 4 zones. The buildings which house the router are - Business Administration C wing (BAC), Old Main, Computing Commons (CPCOM) and Goldwater Center (GWC).

These routers are connected together by Gigabit Ethernet links using Fiber Optic technology in a mesh topology (for providing redundant paths) and this forms the backbone of the network. This is what constitutes the core layer. Figure 12 shows the backbone topology of the ASU campus network. The routers are represented as *router-GWC, router-BAC, router-CPCOM, router-OLDMAIN* according to their respective locations at GWC, BAC, CPCOM and OLDMAIN. The 5th router that shows up is the one that connects to the Internet Service Provider and is a critical part of the backbone. It is named *router-OLDMAIN-INTERNET* as it as also housed in the Old Main building.

Several layer-2 switches are connected to each of these backbone routers that form the distribution layer. From every individual faculty office or research lab in the campus the user plugs his PC or SUN workstation directly or through a hub/switch to those switches in the distribution layer and this forms the access layer. This explains the hierarchical nature

Figure 13. The Building Block

of the system.

3.2.2. Basic Building Block/Module. The campus network is based on a simple building block or module, which is the foundation of modular design. The same building block is used for each building in each region of the campus design. The campus backbone serves to scale from the building model to the campus model. Figure 13 shows the basic building block that is used redundantly to compose the entire campus network. The building block by itself is composite in the sense it can be decomposed into smaller blocks like buildings that constitute of floors and which in turn constitute of labs. Figure 13 shows the generic structure that is present redundantly but there will be variations. Since we have identified such basic blocks these can will be re-used in SESM/CM to build the model.

3.2.3. Main Components of the ASU Computing Network. Arizona State University's campus computer network extensively uses the Cisco product line. With a few

exceptions all the wiring consists of various speeds of ethernet (10Mbps, 100Mbps, Gigabit, and T1s for Internet) and almost all of the hardware consists of Cisco switches and routers. The 2900 series are the mostly used switches for uplinking to 5500 series fast ethernet. (Uplinked - Connection to an uplink port which is an ethernet port that allows network hubs or switches to connect to other hubs or switches without a null-modem, or crossover cable).

The backbone network consists of 5 Cisco Catalyst 5500 backbone routers in a Mesh topology interconnected by a full-duplex gigabit ethernet mesh. The connection is established through fiber-optic cables. Mesh networks can be in two types - *full* or *partial*. A full mesh interconnects all the routers whereas a partial mesh interconnects only some routers. Here our campus backbone uses the full mesh topology since every backbone router is connected to every other router. The full-mesh topology offers many benefits like redundancy and scalability.

The ASU-CCN is comprised of devices and links that connect these devices. The devices that form the backbone are the backbone routers which are Cisco Catalyst 5500 series and are responsible for routing. Catalyst 5000 series switches are attached to these backbone routers and offer connectivity to local sites, which can be administrative services, faculty rooms, research labs, classrooms, student dorms etc. These 5000 series switches are housed in almost all major buildings with high computational needs. Every port of the Catalyst 5000 switch is connected to Layer 2 switches (Catalyst 2900 series) and these 2900 series switches are deployed in the wiring closets for desktop connectivity in the access layer. Table 1 shows the devices and their description.

The links are established through CAT 5 cables and fiber optic. Table 2 shows the links and their description.

Network Device	**Description**
Cisco 5500 Router	Backbone Routers placed in the core layer. Responsible for high speed switching of data
Catalyst 5000 Switch	Gigabit Ethernet Switches placed in the distribution layer
Catalyst 2900 Switch	Deployed in the wiring closets for desktop connectivity to access layer

Table 1. Network Devices and their Description

Network Link	**Description**
Fiber Optic Cable	Used for cabling the campus backbone and between wiring closets if separated by a large distance
Category 5 Cable (CAT 5)	Used for connecting devices to wiring closets and in between wiring closets

Table 2. Network Links and their Description

Desktop connectivity may be established in 2 possible ways - direct connectivity, domain connectivity. When a switch port directly plugs into a PC through a wiring closet it is a direct deployment/connectivity. This is usually the case in faculty rooms. But in most of the research labs, the people in the research group form a domain and may have their own authentication and print server, peripheral devices etc., which belong exclusively to the members of their own group. In such case, they connect to a common hub and the hub uplinks to a port in the Layer 2 switch. This is the case of domain connectivity. The mesh connection between the 5 routers is via high-speed 1000Mbps Fiber-Optic cables. The router-switch connection is done through 100Mbps CAT 5 cables. In case the 5000 series switch that is to be connected is very far from the building that houses the router, then in that case too the connection is established through Fiber-Optic.

Figure 14 shows the entire hierarchical, modular structure of the Arizona State University's multilayer campus network along with the specifics of the devices used. As

Figure 14. ASU Campus Computing Network Diagram

you will see ASU campus network design is one among the many possible designs of a large campus network. The switches in the distribution layer have no direct connectivity. If they have to communicate they have to go through the router that they are attached to. Similarly the devices in the access layer may or may not be connected directly to each other. If they are in the same ethernet LAN they communicate through a Hub. Else if they are in different LAN segments they have to communicate through the immediate upper level Catalyst 2900 switch present in their hierarchy.

CHAPTER 4

SYSTEM MODELING COMPLEXITY MEASURES

4.1. Concepts for measuring model complexity

Software engineering, system-theory, and more generally system sciences and engineering have accounted for various types of structural representations for artificial and natural systems [Bass98, Boo98, DoD99, Elm99, Roz83, Wy93, Zei84]. The two most commonly used modeling primitives for capturing structural representations are "is-a" (i.e., composition) and "part-of" (i.e., specialization) relationships. With these, composite modeling constructs such as mixed composition and specialization are possible. The composition and specialization relationship can have associated cardinalities. For example, a model of a Local Area Network can be composed of 6 personal computers, 2 workstations, a printer, a hub and a scanner. Similarly, a personal computer can be specialized to 2 types - e.g., a desktop and a notebook.

The relationships among components in object-oriented modeling languages such as UML include composition and specialization. These models often also include association and dependency relationships. The modeling constructs offered by UML may be used not only for objects, but also for classifiers. The Scaleable Entity Structure Modeling methodology [Sar03], unlike UML, offers a simpler set of modeling constructs. It supports

is-a and part-of relationships as well as "couplings" among components at the same layer of the hierarchy or between a parent and its children.

Couplings are related to the dependency relationships as defined in UML. They model uni-directional message passing among components using input and output ports [Wy93, Zei00]. That is, couplings specify component interactions via ports. This type of specification is advantageous since it affords defining simple metrics for computing the degree of components interactions. Conceptually, as the number of component and types, ports, and couplings in a model increases, its complexity increases accordingly.

Computing the number of components of a hierarchical model becomes important for non-trivial models. That is, for large-scale models, it becomes impractical to manually compute number of model components, their variants, and their interconnections. For example, the ASU-CCN (refer to Chapter 3) model can have several thousands components and be configured in numerous alternative ways. Therefore, to count the number of switches, routers, computers, and links, and other devices in such a network, it is necessary to define a set of metrics and automate their measurements.

A variety of metrics can be associated with a modular, hierarchical model where components interact with one another through input/output ports and couplings. Given a hierarchical model specified in SESM, we can compute the number of atomic and coupled model components at any given level of its hierarchy, number of input and output ports for atomic and coupled models, number of couplings for coupled models, number of specializations of atomic and coupled models. To provide further details of these basic measurements, it is useful to describe the three complementary SESM modeling structures [Sar03, FU02]. They are

- **Template Model** - A template model can be either an atomic or a coupled model

each with its unique name. Each coupled model can have as its elements a finite number of other template models provided that the depth of each template model is either one or two. The allowed relationships among template models are "has-part" and "kind-of". Every template model is unique structurally w.r.t. all other template models in a given family of models.

- **Instance Template Model** - An instance template model is an instantiation of a set of template models. An instance template model can be an instance of a template model or a synthesis of two or more unique template models. An ITM represent models with arbitrary, but finite, depth. All instances of a template are distinguishable from one another. Each instance template model has a unique ID in addition to what it inherits from TM (structure and unique name). Instance template models can have has-part and kind-of relationships. Since ITM supports representing multi-level hierarchical models, child and grandparent relationship can exist.

- **Instance Model** - An instance model is an instantiation of a template instance model. Each instance model is an exact copy of a template model. All instances of an instance template model are distinguishable from one another. Each instance model has its own ID that can be used to distinguish it from other instance models with the same name.

The Template Model and the Instance Template Model are necessarily the same in terms of the information they represent. But the Instance Template Model can present to the modeler, the multi-level structure of the developed model. It displays the hierarchical model and its children to their deepest level of decomposition. We intend to calculate the metrics associated for each distinct Template Model that is present. In effect it will also

be applicable to the Instance Template Model also since a particular model has the same metrics associated with it irrespective of where and how many times it appears on the hierarchy. As every Template Model is created the metrics associated with it is instantiated and is updated continuously as the Template Model evolves.

The above metrics can be computed based on the information captured in Template models. The same set of metrics, excluding those related to alternative model types, can also be computed for Instance Models.

4.2. Metrics Table

The Metrics table displays the following information relating to every COUPLED model.

- **Name of the Template Model**
- **Type of the Model** (ATOMIC, COUPLED, SPECIALIZED)
- **Number of Immediate Children** This is the Number of immediate or direct children of the Template Model.
- **Number of Total Children** - This is the total number of children in the hierarchy of the Template Model. It is the sum of its children, grandchildren, great grand children and so on till the leaf nodes.
- **Number of Input Ports** - This is the total number of input ports of the Template Model itself.
- **Number of Input Ports** - This is the total number of output ports of the Template Model itself.

- **Total Number of Ports** - In the present version of SESM/CM, this is the sum of the input ports and output ports of the Template Model. This attribute was created with the intention that it will be further extended to display the value of the total number of input and output ports of the Template Model's entire hierarchy till the leaf nodes.

- **Number of Internal Couplings** - This is the total number of internal couplings of the Template Model itself. Internal coupling connects the output ports of the components to input ports of other components.

- **Number of External Input Couplings** - This is the total number of external input couplings of the Template Model itself. The external input coupling connects the input ports of the coupled model to one or more of the input ports of its components.

- **Number of External Output Couplings** - This is the total number of external output couplings of the Template Model itself. The external output coupling connects the output ports of the components to one or more of the output ports of the coupled model.

- **Total Number of Couplings** - In the present version of SESM, this is the sum of the external input couplings, external output couplings and internal couplings of the Template Model. This attribute was created with the intention that it will be further extended to display the value of the total number of couplings of the Template Model's entire hierarchy till the leaf nodes.

The above information is displayed as a Table with the attributes in the left and their corresponding values in the right. For the ATOMIC model the metrics is limited to Name, Model Type, Number of input/output ports when displayed to the user since the number

Template Model - Metrics

Attribute	Value
Model Name	Zone2
Model Type	COUPLED
Children	
Immediate	2
Total	1 4
Ports	
Input	1
Output	1
Total	2
Couplings	
Internal	2
External Input	1
External Output	1
Total	4

Figure 15. Sample of Metrics Associated with a COUPLED Model in SESM/CM

of Children and couplings have no relation to the atomic model. (But the actual database table has all these attributes initialized with null values to aid further expansion of the Model). Figure 15 shows a screenshot of the metrics for a COUPLED model as displayed in SESM/CM.

4.3. SESM Database

4.3.1. Entity Relationship Diagram. Figure 16 shows the Entity Relationship Diagram developed for designing the previous version of SESM with the newly added entities and relationships. In SESM/CM the ER diagram design is extended and the new Metrics entity and its associated relationship are added. This is represented within dotted lines. Table 3 shows the entities and relationships in the ER diagram along with their description. The newly added parts are represented in bold letters.

Figure 16. SESM ER Diagram

Label	**Note**
containsPT	Template Model contains Template Port
containsPTI	Instance Template Model contains Instance Template Port
modelTemplate	Template Model
portTemplate	Port Template
modelTI	Instance Template Model
modelInstance	Instance Model
MTtoMTI	Template Model to Instance Template Model
MTItoMI	Instance Template Model to Instance Model
MTItoSMI	Instance Template Model to Instance Model
PortTI	Instance Template Port
PTtoPTI	Template Port to Instance Template Port
specialized	Template Model can be specialized
componentOf	Decomposition of coupled Template Model
componentOfI	Decomposition of coupled Instance Model
coupling	Coupling between two ports
containsMET	**Template Model contains Metrics**
Metrics	**Metrics generated for the Template Model**

Table 3. ER Diagram Explanation

4.3.1.1. *Added Metrics Entity in SESM Entity-Relationship Diagram.* Along with the previously developed entities (modelTemplate, portTemplate, modelTI, portTI and modelInstance) the current ER diagram also introduces the new Metrics entity.

- Attributes
  - iChildren (Number of Immediate Children for the Template Model)
  - tChildren (Number of Total Children for the Template Model)
  - inP (Number of Input Ports for the Template Model)
  - outP (Number of Output Ports for the Template Model)
  - totP (Total Number of Input/Output Ports)
  - ic (Number of Internal Couplings for the Template Model)
  - eic (Number of External Input Couplings for the Template Model)
  - eoc (Number of External Output Couplings for the Template Model)
  - tc (Total Number of Couplings for the Template Model)
- Descriptions
  - The Metrics entity represents the generated metrics for every Template Model in the SESM specification
  - Metrics is a weak entity of modelTemplate because a metric should not exist if the modelTemplate contains it does not
  - The modelTemplate and Metrics is linked by containsMET (contains Metrics) relationship
  - The attribute inP / outP / totP contain the number of input ports / output / total ports of the modelTemplate

- The attribute eic / eoc / ic / tc contain the number of external input couplings / external output couplings / internal couplings / total couplings of the modelTemplate
- The attribute tChildren / iChildren contain the number of Total / Immediate children of the modelTemplate

4.3.1.2. *Added containsMET Relationship in SESM Entity-Relationship Diagram.* **containsMET** - containsMET defines the relationship between a Template Model and its generated metrics. It is the identifying relationship of the weak entity, Metrics. The relationship is one-to-one between modelTemplate and Metrics. Both the modelTemplate and Metrics have total participation in the relationship. If a modelTemplate exists it should have corresponding Metrics and if Metrics existed it should belong to a modelTemplate.

4.4. UML Data Model Diagram

The previous version of SESM consisted of a database with 10 relational tables (except the Metrics table) based on the ER diagram in the previous section. In this thesis we switch from ER to UML data model to model the physical database of SESM/CM.

The UML offers a standard notation that is very much similar to the ER notation but with additional useful capabilities to produce data models. Using UML the modeler can model not only design database models, but can also reverse engineer an existing database to create its data model diagram. Thereby the modeler can hold a very good representation of not only the structural aspects of his database but he can also have a representation of the behavioural characteristics, which is explained in the following paragraphs. The advantage of using UML data models is that UML is semantically more expressive than ER techniques

[Boo]. This can be jusitified because along with the attributes and relationships UML data model also allows to model the behavior of the database. The behaviors may be

- A primary key constraint (PK)
- A foreign key constraint (FK)
- An index constraint (Index)
- A trigger (Trigger)
- A uniqueness constraint (Unique)
- A stored procedure (Proc)
- A validity check (Check)

This is a very useful representation for our model database because we have employed various constraints in our physical database. For example the type of a model template can either be "ATOMIC", "COUPLED" or "SPECIALIZED". Our implementation should be such that it shows an error if some other data is stored for the model type apart from any of the above three. So we have used a validity check constraint to implement this. Such a situation can be very well modeled using UML rather than ER.

Comparing to the ER diagram described in Section 4.3 that was used to design our database, the UML data model we have created here describes the actual physical implementation of the database. This is the model of the internal database structure. Entities in ER diagram are represented as tables in the data model diagram. These tables represent the actual database tables. Each table can contain columns, constraints triggers and indexes. Tables are joined together through relationships. The values in columns are controlled by constraints.

An identifying relationship between two tables specifies that a child table cannot exist without the parent table [GOR02]. It is a relationship of part-to-whole, where the part would have no sense without the whole. When we use an identifying relationship the primary key of the parent table migrates to the child table as primary/foreign key. The foreign key then is part of the child table's primary key constraint and foreign key constraint. This enforces referential integrity of the model, making the child table's existence dependent on the parent table's existence. This referential integrity prevents orphan records in the child table by requiring the deletion of the instance in the child table first, before deleting the instance in the parent table.

In our data model diagram the MODELTEMPLATE table represents a Template Model in SESM. All attributes are single-valued and not nullable. The attribute TID (modelTemplate name) is the primary key, which is denoted as PK. The PORTTEMPLATE table represents the ports that belong to Template Models. The portTemplate by itself has no meaning unless it is related to a modelTemplate. The portTemplate should not exist if the Model Template that contains it does not. In our ER diagram the PORTTEMPLATE is represented as a weak entity of MODELTEMPLATE. The corresponding relationship in our data model diagram is an identifying relationship between MODELTEMPLATE and PORTTEMPLATE.

Figure 17 shows (a part extracted from from Figure 20) the relationship between MODELTEMPLATE and PORTTEMPLATE tables. The primary key TID of MODELTEMPLATE migrates as the foreign key OWNER in portTemplate and also acts as the primary key of portTemplate.

Non-identifying relationships are relationships in which there is no strong interdependency between the child and parent instances , hence the foreign key is not embedded

Figure 17. An Identifying Relationship Example

in the child table's primary key constraint [GOR02]. A non-identifying relationship specifies a regular association between two independent tables. A data model non-identifing relationship does not require the foreign key to be part of the child table's primary key. In our data model, there is no non-identifying relationship, since all the tables are strongly dependent on the existence of the ModelTemplate table. In our data model diagram we do not have any non-identifying relationships since every table is fully dependent on the existence of the Template Model table. Cardinality represents the minimum and maximum number of possible instantiations of a relationship between two tables. Cardinality is used to enforce referential integrity. If a table has a cardinality of 1, then that signifies the table must exist in the relationship. This cardinality is especially important for parent tables to prevent orphan records in the child tables. Cardinality can also determine if a foreign key is unique and can be nullable. Table 4 specifies the necessary cardinalities to make a foreign key nullable and/or unique.

The MODELTEMPLATE table has a cardinality of 1:0..* with the PORTTEMPLATE table . This implies that the foreign key is not nullable and also need not be unique. We use this relationship here since each ModelTemplate can have many ports

Required Constraints	**Parent Table Cardinality**	**Child Table Cardinality**
Foreign Key is nullable and unique	0..1	0..1 or 1
Foreign Key is nullable and not unique	0..1	0..* or 1..*
Foreign Key is not nullable and not unique	1	1..* or 0..*
Foreign Key is not nullable but is unique	1	0..1 or 1

Table 4. Required Table Constraints and their Cardinality

and hence the modelTemplate name will appear multiple times in the portTemplate table depending on the number of ports.

An intersection table is a relationship structure in which one child table is related to two parent tables with two 1:N identifying relationships [GOR02]. When such a structure is created the child table contains primary/foreign keys corresponding to the parent table's primary keys. Therefore any updates made to the child table will affect both parent tables. The COMPONENTOF table is an intersection table for MODELTEMPLATE and MODELTI tables. It is connected to the MODELTEMPLATE and MODELTI with a 1:0..* cardinality. The primary keys of MODELTEMPLATE and MODELTI have migrated as the primary/foreign keys of the COMPONENTOF table. Whenever a component is added, the COMPONENTOF table has an added row. This also performs an SQL UPDATE on the MODELTEMPLATE table and an SQL INSERT on the MODELTI table. So, this table is an intersection table. Figure 18 shows the componentOf and its relationship with the ModelTemplate and ModelTemplateInstance.

The MODELTEMPLATE table has a one-to-one cardinality relationship with METRICS table. From Table 4 we can infer that one-to-one relationship is the case when foreign key is not nullable but is unique. Since we have single collective metrics data for each and every modelTemplate, the modelTemplate's TID appears one and only once in the

Figure 18. An Intersection Table Example

Figure 19. Relationship between Metrics table and ModelTemplate table

METRICS table's foreign key (as attribute NAME) and the value cannot be null at any point of time (since metrics for every model is initialized as soon as the model is created). The attribute NAME in the Metrics table is the primary key and is also the foreign key derived from the TID of the ModelTemplate table. Also these two tables are related by an identifying relationship as a primary key in the Metrics table should not exist without the corresponding key existing in the ModelTemplate table (See Figure 19). Figure 20 and Figure 21 show the complete UML data model view of all the tables in SESM/CM. The UML data model diagram is represented in 2 views for clarity.

Figure 20. Data Model Diagram of SESM/CM Database - View 1

Figure 21. Data Model Diagram of SESM/CM Database - View 2

CHAPTER 5

DESIGN AND SPECIFICATION OF SESM

5.1. Design Requirements

As defined in Chapter 2 the architecture of SESM may be decomposed into four parts. They are the SESM server, SESM client, the network environment and the relational database. The architecture is designed such that the client can establish a direct connection to the database but can perform only read operations. If the client has to write to the database it has to go through the server. Figure 23 illustrates the concept.

SESM client contains graphical user interface and the logics to support user interactions (send and receive) with the server via the network environment. The GUI Class utilizes the SESM query component to retrieve data from the DBMS and map the relational models into their object-oriented counterparts. The client can either receive a notification or a request for reply from the server. If a notification is received, the client refreshes the GUI accordingly and notifies the user about the modifications made. If a request for reply from the server was received, the client interacts with the user to obtain the reply and send it back to the server. The graphical user interface (GUI) has three major functionalities: displaying the hierarchical structure of models, displaying details of models, and allowing the user to modify models.

Figure 22. SESM Server Use Case Diagram

Having described the existing architecture this chapter describes the focus of this thesis, which is computing Complexity Metrics for the models. To accomplish this the existing architecture of SESM was extended. Modifications and further extensions were made to the existing client, server and DBMS (See Figure 23) to incorporate Metrics computation into the system. The existing architecture is good as such so our concern was not to re-design the system. We did not make any structural changes to the system but extended the architecture at the code level. As said earlier that modifications and extensions were made to the client, server and relational database. The following sections describe each one of them in detail. The SESM server allows a user to perform add, delete and modify operations to the models. These operations involve multiple transactions to the relational database. Figure 22 shows the SESM Server use case diagram.

5.2. Extensions to the Relational Database

The previous version of SESM the database consisted of 10 relational tables:

1. ModelTemplate
2. PortTemplate
3. Specialization
4. ModelTI
5. ComponentOf
6. PortTI
7. Coupling
8. ModelInstance
9. MTITOSMI
10. ComponentOfI

A new relational table named Metrics is added based on the revised ER diagram and the UML data model diagram. The relational database schema specification of the Metrics table for incorporating complexity measure is as given in Table 5. Foreign Keys are shown as ***bold-italic*** and Primary Keys are shown as **bold**. All other column names are shown in plain font.

name is a foreign key from modelTemplate (tID)

tModelType is a foreign key from modelTemplate (tModelType)

numChildren is an integer

totalChildren is an integer

inPorts is an integer

outPorts is an integer

totalPorts is an integer

Metrics										
name	tModelType	numChildren	totalChildren	inPorts	outPorts	totalPorts	ic	eic	eoc	tc

Table 5. Relational Database Schema Specification for Metrics Table

ic is an integer

eic is an integer

eoc is an integer

tc is an integer

The primary key is name

Below is the Data Definition Language or the SQL statement that defines the metrics table.

```
CREATE TABLE METRICS (NAMEVARCHAR2 (100), TMODELTYPE VARCHAR2 (13),
NUMCHILDREN NUMBER (25), TOTALCHILDREN NUMBER (25), INPORTS NUMBER (25),
OUTPORTS NUMBER (25), TOTALPORTS NUMBER (25), EIC NUMBER (25), EOC NUMBER
(25), IC NUMBER (25), TC NUMBER (25), PRIMARY KEY (NAME), FOREIGN KEY
(NAME) REFERENCES MODELTEMPLATE (TID) ON DELETE CASCADE )
```

5.3. Extensions made to SESM Server

As we know, SESM server allows a user to perform add, delete and modify operations to the models. These operations are sent to the DBMS as data manipulation transactions. Whenever an operation is made on a Template Model, the metrics may change. This new value is constantly updated in the Metrics table through data manipulation transactions.

Figure 23. SESM/CM Architecture - Parts Modified

The types of transaction can be SQL INSERT, SQL DELETE or SQL UPDATE. Examples of each SQL statement with the specific key words are shown below.

- SQL INSERT: INSERT INTO METRICS VALUES ('ROUTER', 'ATOMIC', 0, 0, 0, 0, 0, 0, 0)
- SQL DELETE: DELETE METRICS WHERE NAME='ROUTER'
- SQL UPDATE: UPDATE METRICS SET TMODELTYPE = 'COUPLED' WHERE NAME = 'ROUTER'

Figure 24 shows the use case diagram for the metrics transactions. The marked portion in the figure denotes the the extension that is made for complexity metrics manipulation.

When an add operation is performed on a Template Model /its component / or its port, the operation is carried out by either an add or a modify transaction in the database. For example when a Template Model is created a new entry should be added in the Metrics table (SQL INSERT). But when a port is added to a Template Model the port count for that

Template Model should be incremented in the Metrics Table (SQL UPDATE). Similarly delete operations involve delete or modify transactions.

This section shows the possible Transactions that are made to a Template Model and the corresponding updates done to its Metrics.

5.3.1. Add Operations (Add / Modify Transactions). Add / modify transactions may occur during any of the following operations.

- When a Template Model is created
- When a Component is added to a Template Model
- When an Port is added to a Template Model
- When a Coupling is added between 2 Template Models

Looking at each of these in detail,

1. When a Template Model is created
    - Input
        - tID - Template Model Name
    - ADD a row in the Metrics table with the following column values
        (a) Name = name of the Template Model (tID)
        (b) ModelType = ATOMIC
        (c) NumChildren = 0
        (d) TotalChildren = 0

Figure 24. Metrics Use Case Diagram

(e) InPorts = 0

(f) OutPorts = 0

(g) TotalPorts = 0

(h) IC = 0

(i) EIC = 0

(j) EOC = 0

(k) TC = 0

2. When a Component is added to a Template Model

- Input
  - tID - Coupled Template Model ID
  - compTID - Component's Template Model ID
- If the Template Model tID's type is atomic in METRICS table change the ModelType to COUPLED
- Check if the child compTID is ATOMIC or COUPLED
  - If the child compTID is ATOMIC then increment the NumChildren and TotalChildren of the Parent Template Model by 1.
  - Else if the child compTID is COUPLED then query Metrics table to find the TotalChildren of the child, say n (where, n = 1..infinity). Update TotalChildren = TotalChildren + n for the Parent Template Model.
  - Query the COMPONENTOF table to check if the Parent Model Template tID is itself a child of another Model Template.
    (a) If the Model Template tID is a child of 1 or more Model Templates, then for each of its parent recursively query the COMPONENTOF table to

find the grandparent and so on till each such found Model Template is not contained in any other Model Template.

(b) ii) For the list of Model Templates found, update TotalChildren = TotalChildren + n or TotalChildren = TotalChildren + 1 depending upon whether the child is ATOMIC or COUPLED.

3. When a Port is added to a Template Model

   - Input
     - Template Model Name - tID
     - Port Name
     - Port Type
   - Check to see if the type of the added port is "IN" or "OUT"
     - If type = "IN" update InPort = InPort + 1 for the corresponding row of ModelTemplate in the Metrics Table.
     - If type = "OUT" update OutPort = OutPort + 1 for the corresponding row of ModelTemplate in the Metrics Table.

4. When a Coupling is added between 2 Template Models

   - Coupling between models can be of two types
     - Between a Coupled Model and its component. In this case it is always either IN port - IN port (External Input Coupling) or OUT port - OUT port (External Output Coupling)
     - Between 2 components of a coupled model. In this case it is always IN port - OUT port (Internal Coupling)

- Check the coupling to find the value of tiIDf (Instance ID)
  - If tiIDf = 0 the coupling is between a coupled Model and its component
  - If tiIDf = 0 and pTypef = "IN" update EIC = EIC + 1 for the corresponding row of ModelTemplate in the Metrics Table
  - If tiIDf = 0 and pTypef = "OUT" update EOC = EOC + 1 for the corresponding row of ModelTemplate in the Metrics Table
- If tiIDf != 0 it means that the coupling is between a 2 components of a coupled Model
  - Query the COMPONENTOF table to find the Owner of the components
  - Update IC = IC + 1 for the Owner in the Metrics table

5.3.2. Delete Operations (Delete / Modify Transactions). Delete / modify transactions may occur during any of the following operations.

- When a Template Model is deleted
- When a Component is deleted from a Coupled Template Model
- When an Port is deleted from a Template Model
- When a Coupling is deleted between 2 Template Models

Looking at each of these in detail,

1. When a Template Model is deleted
   - Input
     - tID - Template Model Name

- Query the MODELTEMPLATE table to find if the model to be deleted is ATOMIC or COUPLED
  (a) If the model is COUPLED query the METRICS table to find its TOTALCHILDREN, say tChildren = n
  (b) If it is an ATOMIC model set tChildren = 0
- Query the COMPONENTOF table to find all the immediate parents of the model to be deleted
- If there are any immediate parents, for all the immediate parents decrement NumChildren by 1, i.e., update NumChildren = NumChildren - 1
- For all the immediate parents found, recursively query the COMPONENTOF table to find the grandparent and so on till each such found Model Template is not contained in any other Model Template.
- For the list of immediate parents and the grandparent hierarchy obtained, update TotalChildren = TotalChildren - (tChildren + 1), where (tChildren + 1) denotes decrementing the count for the number of children of the model to be deleted plus the model itself.
- DELETE the row in the METRICS table which corresponds to tID, i.e., the model to be deleted

2. When a Component is deleted from a coupled Template Model

- Input
  – tID - Coupled Template Model ID
  – compTID - Component's Template Model ID

  - compTIID - Component's Instance Template Model ID

- Query the MODELTEMPLATE table to find if the model to be deleted (compTID) is ATOMIC or COUPLED
  - If compTID is COUPLED query the METRICS table to find its TOTALCHILDREN, say tChildren = n
  - If it is an ATOMIC model set tChildren = 0
  - For the direct parent tID update NumChildren = NumChildren - 1, TotalChildren = TotalChildren - (tChildren + 1)
  - Recursively query the COMPONENTOF table to find all the immediate parents and grandparents and so on for tID.
  - For the list obtained update TotalChildren = TotalChildren - (tChildren + 1)
  - If the deleted component is the only child of the coupled model tID, then change the ModelType to ATOMIC for tID in the METRICS table

3. When a Port is deleted from a Template Model

   - Check to see if the type of the added port is "IN" or "OUT"
     - If type = "IN" update InPort = InPort - 1 for the corresponding row of ModelTemplate in the Metrics Table
     - If type = "OUT" update OutPort = OutPort - 1 for the corresponding row of ModelTemplate in the Metrics Table

4. When a coupling is deleted between 2 Template Models

   - Check the coupling to find out the value of tiIDf

- If tiIDf = 0 the coupling to be deleted is between a coupled Model and its component
  - If tiIDf = 0 and pTypef = "IN" update EIC = EIC - 1 for the corresponding row of ModelTemplate in the Metrics Table
  - If tiIDf = 0 and pTypef = "OUT" update EOC = EOC - 1 for the corresponding row of ModelTemplate in the Metrics Table
- If tiIDf != 0 it means that the coupling is between 2 components of a coupled Model
  - Query the COMPONENTOF table to find the Owner of the components
  - Update IC = IC - 1 for the Owner in the Metrics table

The transactions performed for delete port, delete coupling, delete component are symmetrical to that of their add counterparts except that decrement operation is carried out instead of increment on the corresponding counts. But deleting a Template Model varies considerably from creating a Template Model. The Template Model that is being deleted may be a child for multiple other Template Models, so those Template Models should be updated in terms of their immediate children and total children count.

5.3.3. Extended and Newly Added Methods: Figure 25 shows the class specification for sesmServer. The following methods have been extended.

- addModelTemplate()
- addPort()
- addComponent()
- addSpecialization()

Figure 25. SESM Server Component Digram

- addCoupling()
- delModelTemplate()
- delPort()
- delComponent()
- delCoupling()
- modifyModelName()

These call the appropriate newly added methods present below.

incrementTotalChildren (String tID, int n) Increments total children count in the Metrics table for the Template Model with Name tID by n. Returns the boolean value true if operation is successful.

incrementNumChildren (String tID, int n) Increments Immediate Children Count in the Metrics table for the Template Model with Name tID by n. Returns the Boolean value true if operation is successful.

decrementTotalChildren (String tID, int n) Decrements Total Children Count in the Metrics table for the Template Model with Name tID by n. Returns the Boolean value true if operation is successful.

decrementTotalChildren (String tID, int n) Decrements Immediate Children Count in the Metrics table for the Template Model with Name tID by n. Returns the Boolean value true if operation is successful.

modelTypeMetrics (String templateID, String modelType) Changes the Modeltype of the Template Model with Name templateID to modelType in the Metrics Table. The modelType may be ATOMIC or COUPLED. This is performed when a component is added to an ATOMIC model (the modelType changes to COUPLED) or the only child is deleted from a COUPLED model (the modelType changes to ATOMIC).

As a sample, the *Add Component* operation is explained in detail. Following are the algorithm and sequence diagram for the Add Component operation.

```
addComponent(String tID, String compID, num)
{

        /*Query ModelTemplate table to find the ModelType of tID */
         tIDmodelType = Query[modeltype(tID)]

        if (tIDmodelType = SPECIALIZED)
                Error: Cannot add component to Specialized Model

         if (loop[tID, compID])
                Error: Loop in the hierarchy

         Add num number of compIDs to parent tID

        /*Update Immediate and Total Children of tID*/
         incrementNumChildren (tID, num)
        incrementTotalChildren(tID, num)

         /*Query ModelTemplate table to find the ModelType of compID */
         compIDmodelType = Query[modeltype(compID)]

         /*If the added child itself is coupled then update tID's Totalchildren to include
        TotalChildren of compID*/
         if (compIDType = COUPLED)
         {
                totalChildren = Query[Metrics(compID)]
                 incrementTotalChildren(tID, totalChildren * num)
         }

         /*For all parents of tID update Totalchildren*/
         Vector owner = Query[componentOf(TOWNER, tID)]
        int n = owner.size()
        for (i=0; i<owner.size;i++)
         {
                String currentElement = owner.elementAt(i)
                Vector newAddition = Query(TOWNER, currentElement)
                for (i=0; i<newAddition.size(); i++)
                {
                        owner.addElement(newAddition.elementAt(i))
                }
                incrementTotalChildren(currentElement, totalChildren * num)
                n = owner.size()
        }

        /*If tID is ATOMIC change the type to COUPLED in Metrics table*/
        if (tIDmodelType = ATOMIC) modelTypeMetrics(tID, COUPLED)

}
```

Figure 26. Algorithm for Add Component Operation

The Add Component operation occurs when a component (or a number of components) is added to an existing template model. The algorithm explains the method *addComponent* that is responsible for this computation. *addComponent* takes 3 arguments.

- tID - Parent Template to which the component is added
- compID - Child component (it may either be ATOMIC or COUPLED) which is added to tID
- num - number of compIDs that need to be added to tID

Figure 27 shows the sequence diagram for the metrics manipulation for an add component operation. It is important to indicate that consistency is preserved in the process of computing metrics. Since we have metrics information stored as a separate tuple for every model in the metrics database table the server posts all changes of a particular model to its corresponding unique row in the database thus maintaining consistency while computing and storing and viewing the metrics. Modifications done to a model are updated at every level of hierarchy the model is present. Figure 28 illustrates this concept.

5.4. Extensions made to SESM Client

Compared to the DBMS and server there are less extensions made to the SESM client. When a user requests to view the metrics for a particular model SESM displays the metrics to the user in a table format. The client is made to perform this operation. When the user requests for viewing the metrics, the client connects to the DBMS (since client

Figure 27. Add Component Operation on SESM Server Sequence Diagram

Figure 28. Metrics Consistency and Uniformity

is allowed to perform read-only operations) and gathers current data from the already populated Metrics table. During this transaction there are NO CHANGES made to the Metrics table. Figure 29 shows the sequence diagram for this operation.

5.4.1. User Interface Enhancements in SESM Client. In the client side, the SESM user interface was enhanced to incorporate mechanism to allow SESM's users view the model complexity metrics. The Graphical User Interface of SESM is explained in Chapter 2. As shown in Figure 3 the model menu is enhanced with an added menu item "Show Metrics". This section explains the Show Metrics option in detail.

Right-click on a Template Model or an Instance Template Model gives a menu that allows users to add or delete component, rename template model and add port. The current version of SESM has been modified to include an option "Show Metrics" for the user to conveniently view the complexity metrics of the Model Template. When the user clicks on

Figure 29. SESM Client Sequence Diagram

Template Model - Metrics

Attribute	Value
Model Name	Zone2
Model Type	COUPLED
Children	
Immediate	2
Total	14
Ports	
Input	1
Output	1
Total	2
Couplings	
Internal	2
External Input	1
External Output	1
Total	4

Figure 30. GUI representing Complexity Metrics for COUPLED Model

Template Model - Metrics

Attribute	Value
Model Name	Hub
Model Type	ATOMIC
Ports	
Input	6
Output	6

Figure 31. GUI representing Complexity Metrics for ATOMIC Model

"Show Metrics" a table as shown in Figure 30 is displayed. Figure 30 shows the Metrics for a COUPLED model. The attributes in the table correspond to the columns in the *Metrics* database table as discussed in section 5.2. However in the case of an ATOMIC model information about Children and Couplings are not pertinent. So, for ATOMIC models SESM/CM limits the display to model name, model type and number of input/output ports (even though all the attributes exist in the Metrics database table for further expansion of the Model). Figure 31 shows the metrics as displayed in SESM/CM for an atomic model.

CHAPTER 6

RESULTS, FUTURE WORK AND CONCLUSIONS

6.1. ASU Campus Computing Network Model in SESM

This section provides samples of the ASU-CCN model created using SESM/CM. Also samples of how the metrics computation is verified are presented.

Figure 32 shows the SESM/CM graphical user interface of the Template Model view for the large-scale model of the ASU-CCN. The Template Model view shows the various components developed to create the model. Figure 39 shows the parts of the University Network expanded in the Instance Template Model view. Figure 33 shows the sub-model University Network which is the root model that spawns the entire hierarchical structure. We have modeled two zones out the four existing zones discussed in the case study in Chapter 3. Figure 33 shows that the COUPLED model University Network has two zones (Zone 1 and Zone 2). These two zones are interlinked to each other in a mesh fashion through corresponding uplink ports. Additionally Zone 2 is connected to the Router-OLDMAIN-INTERNET. The Router-OLDMAIN-INTERNET provides interconnectivity service to ASU by connecting to the Internet Service Providers (ISP). The ISP sub-model has input/output ports for AT/T, Sprint and Qwest. We can examine details of the Zone 1 in terms of its components and connections (i.e., couplings). As explained in Chapter 3 every zone has a Router. Figure

Figure 32. Large-Scale Model of the ASU Campus Computing Network in SESM/CM

Figure 33. University Network SubModel

Figure 34. Zone1 SubModel

34 shows Zone 1 with the Router-GWC component. We have modeled 2 networks in Zone 1 - GWC Network and ECG Network components. These two network components are individually connected to the ports of the Router-GWC component. Examination of the internals of the GWC Network in Zone 1 shows it has a Cisco 5000 Series switch and 3 Floors (see Figure 35). Floor 1 is ETS, Floors 2 is CSE-Floor2 and Floor 3 is CSE-Floor3, each of which is a coupled model. The Cisco 5000 Series switch is an atomic model having three input and three output ethernet ports (for receiving/sending connections from/to every floor) and a pair of Uplink input and output ports (for connecting to a 5000 Series Switch in another building Network). Figure 36 shows CSE-Floor3 of the GWC Network.

Figure 35. GWC Network SubModel

Figure 36. Floor 3 of the GWC Network

Floor 3 has a Wiring room, the ACIMS Lab and another Research Lab (Research Lab1). The Hub/Switch ports from the Labs connect to the Switch ports in the Wiring Room.

This part describes two samples of verification of our implementation of the complexity metrics. Figure 37 shows Zone 1 and its corresponding metrics table as viewed by the modeler. The table shows Zone 1 as a COUPLED model with 3 Immediate Children (Router-GWC, GWC Network and ECG Network) and 86 Total Children. It also depicts that Zone 1 has 1 Input Port, 1 Output Port, and 4 Internal Couplings, 1 External Input Coupling and 1 External Output Coupling. Figure 38 shows sample metrics for the ACIMS laboratory.

Attribute	Value
Model Name	Zone1
Model Type	COUPLED
Children	
Immediate	3
Total	86
Ports	
Input	1
Output	1
Total	2
Couplings	
Internal	4
External Input	1
External Output	1
Total	6

Figure 37. Metrics for Zone1

Figure 38. Metrics for ACIMS Lab

Figure 39. Instance Template Model View of University Network

Figure 40. A Possible Extension to the Couplings Metric

6.2. Future Work

6.2.1. Computation of Additional Metrics. SESM/CM facilitates computation of nine basic structural metrics - (Immediate Children, Total Children, Input Ports, Output Ports, Total Ports, Internal Couplings, External Input Couplings, External Output Couplings, Total Couplings). However, additional structural metrics may be obtained from the models. For example, higher-level metrics such as fan-in (the number of couplings between an input port and many output ports) and fan-out (the number of couplings between an output port and many input ports) can be computed. Also the present metrics compute the number of couplings at one level. This may be extended, for example, to compute the total number of couplings of a component till its lowest level of decomposition as shown in Figure 40.

6.2.2. Support for Collaborative Model Development. The SESM environment is developed primarily for a single user. Higher-level capabilities such as keeping track

of the modelers' identity, joining/leaving a modeling session, and coordination of clients' activities (e.g., ensuring correct sequencing of events such as adding a component, deleting a coupling, etc.) are not supported by SESM. This suggests a new architecture design that uses collaborative modeling techniques [Sar99] and a suitable middleware such as Collaborative Distributed Network System [Par98, Sar00] instead of the network environment.

6.2.3. Support for storing Multiple Models. SESM/CM currently supports the development of a single large model with multiple related sub-components. Existence of multiple independent models using a single user account is not supported by the SESM/CM. Therefore, it is useful to support storage and retrieval of multiple distinct models by a single user. In addition, it is desirable to support reuse of components among different models.

6.3. Conclusions

Computer modeling techniques are extensively used to characterize various structures of large-scale dynamical systems. Visual depiction of large-scale models with complex structures is non-trivial since it needs to support comprehension and understandability of design choices. It is, therefore, key to represent the complex structures both in graphical and logical form. Furthermore, as described earlier, it is important to determine, in a quantitative fashion, a model's complexity using metrics. These metrics can be measured and used as an aid to evaluate alternative design choices. For example, during the architectural design of a component-based system, complexity measures can serve as one metric to select one design vs. another.

To our knowledge, no modeling environment (both general modeling tools like UML or domain specific tools like OPNET) exists that that can support capturing and comput-

ing complexity measures of the modular, hierarchical models. This resulted in the software development of the SESM/CM - i.e., support for capturing and updating quantitative structural metric information such as the number of components and couplings of a model as it evolves, stored persistently and made available to modelers.

As discussed earlier, using the SESM/CM, in addition to composition, specialization can also be modeled and measured. We can count the number of specializations for individual atomic components and across the entire model. Such metrics, therefore, offer added capability to choose among competing design choices. Therefore, metrics based on structural designs (i.e., composition and specialization) allows examination of the details of design decisions.

This research addresses the importance of structural quantitative measurements through the development and implementation of a set of metrics computation for the models created through an existing modeling environment Scalable System Entity Structure Modeler (SESM). In this thesis, we described the development of the SESM/CM software design which is an extension of the SESM to capture and compute persistent complexity metrics for modular hierarchical models. To implement this, we studied the existing architecture of the SESM and made extensions to the SESM's server, client and database modules to incorporate capturing of metrics and its persistent storage. The overall design of the architecture of SESM was retained. Our main intention was to extend the SESM environment to be able to compute basic structural complexity metrics. This entailed incorporating extensions to the SESM's server, client and database which resulted in SESM/CM. We have modified the SESM Server to support representation and computation of metrics such that structural measurements are dynamically computed and updated as the model changes during modeling sessions. A model's complexity measures are confined to a separate table in the

SESM/CM database. Finally the SESM Client has been enhanced to display the metrics of any model in a tabular form whenever requested by the user.

A case study of the Arizona State University Campus Computing Network (ASU-CCN) provided a valuable test case. The interactions and interviews with the Computer Networking experts in the IT department at ASU played a key role in the understanding of the computer network design choices and the subsequent development of its conceptual model. This model was important in the development of the logical model of the computing network. Thorough testing was carried out to verify the correctness of the detailed design and the implementation of the complexity metrics computation using the ASU-CCN model. An important part of the SESM/CM software implementation was to ensure consistency of the complexity measures across the entire model.

We demonstrated the SESM/CM tool to the ASU networking experts. The feedback from the experts showed the usefulness of the metrics in the better understanding of the ASU-CCN network and its value for its maintenance. For instance the total number of children of the parent model of a switch is an indicator of the total number of nodes a switch interconnects. This kind of information, for example, allows the university system administrators to weigh alternative design choices given available bandwidth for making plans to support higher demands.

The graphical view of the campus network model provided a user-friendly medium for developing and communicating design alternatives. Finally, we discussed some important features for extending the capabilities of the SESM/CM environment including additional computational metrics, support for multiple model storage and reuse, and collaborative modeling.

REFERENCES

[Asu03] ASU Map http://www.asu.edu/map/map.html/.

[Bass98] Bass, L., P. Clements, and R. Kazman, *Software Architecture in Practice. The SEI Series in Software Engineering*. 1998: Addison Wesley.

[Boo] Booch, G., *Unifying Enterprise Development Teams with the UML*. Rational Software White Paper

[Boo94] Booch, G., *Object-Oriented Design with Applications*. 1994, Redwood City, CA: Benjamin/Cummings.

[Boo98] Booch, G., I. Jacobson, and J. Rumbaugh, *The Unified Modeling Language User Guide*. 1998: Addison-Wesley. 482.

[Bry98] Bryan, D., *Using Rapide to model and specify inter-object behavior*. 1994, Stanford University.

[DoD99] DoD, U.S., *High-Level Architecture Object Model Specification* (Version 1.4). 1999.

[Elm99] Elmqvist, H., et al., *Modelica(TM): A Unified Object-Oriented Language for Physical Systems Modeling*. 1999,http://www.modelica.org/. p. 1-49.

[FU02] Fu, T.-S., Hierarchical Modeling of Large-Scale Systems Using Relational Databases, Electrical and Computer Engineering, University of Arizona, Tucson, 2002.

[GOR02] Gornik, D., *UML Data Modeling Profile*. Rational Software White Paper, May 2002.

[Har87] Harel, D., *Statecharts: A Visual Formalism for Complex Systems*. Science of Computer Programming, 1987. 8(3): p. 231–274.

[Mat98] Mathworks *MATLAB*. 1998, http://www.mathworks.com/.

[Mod00] Modelica Association *Modelica*. 2000, http://www.modelica.org/.

[Opn] OPNET Technologies, Inc., *OPNET* http://www.opnet.com/.

[Ora03] RDBMS http://technet.oracle.com/.

[Par98] Park, S., *Collaborative distributed newtork System Architecture: Design and Implementation*. 1998, MS Thesis, Electrical and Computer Engineering, University of Arizona: Tucson, AZ, USA

[Pre97] Pressman, R., *Software Engineering: A Practitioner's Approach*. Fourth Edition ed. 1997: McGraw Hill.

[Ree00] Reese, G., *Database Programming with JBBC and Java*. 2nd Edition, O'Reilly and Associates, 2000.

[Rep03] Repp, J., *Personal Comunication*

[Ros00] Rational, *Rational Rose*. 2000: http://www.rational.com./.

[Roz83] Rozenblit, J.R. and B.P. Zeigler. *Representing and Construction System Specifications Using the System Entity Structure Concepts.*Winter Simulation Conference. 1983. Los Angeles.

[Sar99] Sarjoughian, H.S., J.Nutaro, et al. *Collaborative Distributed Modeler*. 1999, SCS International Conference on Web-based modeling and simulation, pp. 119-124, SCS, January, San Francisco, CA.

[Sar00] Sarjoughian, H.S. and S.Park *Collaborative Distributed Network System: A Lightweight Middleware Supporting Collaborative Distributive Modeling*. 2000, Future Generation Computer Systems 710, Vol. 17, pp.89-105.

[Sar03] Sarjoughian, H.S., Fu, T-S., *An Approach for Scaleable Model Representation and Management*. 2003, Computer Science and Engr., Arizona State University, Tempe, Arizona, in Preparation.

[Tho01] Thomas M.T. and and Doris E. Pavlichek, *Cisco Internetwork Design*. McGraw-Hill, 2001.

[UML00] UML *Unified Modeling Language*. 2000:http://www.rational.com/uml/index.jtmpl.

[Wy93] Wymore, A.W., *Model-based Systems Engineering: An Introduction to the Mathematical Theory of Discrete Systems and to the Tricotyledon Theory of System Design.* 1993, Boca Raton: CRC.

[Zei00] Zeigler, B.P., H. Praehofer, and T.G. Kim, *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Second Edition ed. 2000: Academic Press.

[Zei84] Zeigler, B.P., *Multi-Facetted Modeling and Discrete Event Simulation.* 1984, New York: Academic Press.