

© 2014 Shashank Chaudhry.

DISCRETE EVENT SIMULATION BASED STUDY OF A PLM SOFTWARE
NETWORK

BY

SHASHANK CHAUDHRY

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Advisers:

Professor Debasish Dutta
Doctor Lalit Patil

Abstract

A model for describing the data storage and data flow in a Product Lifecycle Management (PLM) software network has been developed in this study. Discrete event simulation (DES) has been used to represent this flow of data and queuing models have been used to represent the hardware components of the network. The aim of the study has been to accurately represent the true behavior of the network during heavy operational load and to develop and run scenarios to increase software user productivity. A flexible tool has been developed for this purpose, which makes it simple to test numerous scenarios and carry out bottleneck analyses. The tool uses three tiers of code to accurately model the PLM software network, the hardware tier, the software tier and the application tier, which includes business insights that are being simulated. The studies show that increasing the cache size in the network can improve the response times only to a certain extent despite significant increase in the number of cache hits and that the response time can be negatively impacted if the number of non-critical users in the network increase. We observe that rather than expensive hardware upgrades, which show modest improvement in response time, to the tune of 5 – 8%, other more innovative business oriented approaches such as allowing the users with deadlines to start work earlier or allowing the cache to be preloaded show much more promising results, with as much as 50% decrease in response time. Statistical validation of the model and the output results has also been carried out.

To my family.

Acknowledgments

I would like to express my sincere gratitude to my advisors, Drs. Lalit Patil and Debasish Dutta, who were always there to guide and help me especially in times of great confusion.

I would also like to thank Krishna Murthy from BIMCON Inc. for his insights and detailed subject matter expertise that was instrumental in ensuring that the work remains relevant to a large manufacturing company.

I believe that the support of my family and friends has also helped me keep a level head during the various stages of my research work and for this, I would like to thank them profusely.

Table of Contents

List of Tables	vii
List of Figures	viii
List of Abbreviations	ix
Chapter 1 Introduction	1
1.1 Product Lifecycle Management	1
1.1.1 Definition	1
1.1.2 Functions of PLM	2
1.1.3 PLM Software	2
1.2 Motivation	3
1.3 Related Work	4
1.3.1 Analytic Approaches	4
1.3.2 Experimental Approaches	4
1.3.3 Simulation Approaches	5
1.3.4 Reasons for using Simulation	5
1.4 Research Contributions	5
Chapter 2 DES – Theory and Implementation	7
2.1 Defining DES	8
2.1.1 Continuous and Discrete Systems	8
2.1.2 Systems and System Events	8
2.2 DES Model Paradigms	9
2.2.1 Activity-Oriented Paradigm	9
2.2.2 Event-Oriented Paradigm	10
2.2.3 Process-Oriented Paradigm	10
2.3 Framework of a DES Study	10
2.4 DES using SimPy	13
Chapter 3 Network Topology of PLM Software	15
3.1 Single-User Network Architecture	15
3.1.1 Formal Description of the PLM Network Topology	16
3.2 Multi-User Network Architecture	17
Chapter 4 Modeling Methodology	19
4.1 Concept of Milestone	19
4.2 Hardware Modeling	20
4.2.1 LFC and FSC as LRU Caches	22
4.3 Software Modeling	24
4.4 Business Process Modeling	25
4.5 Simulation Framework	29

Chapter 5	Results and Discussion	33
5.1	Testing the Effect of FSC Cache Size	33
5.2	Testing the Effect of Increasing Non-Milestone Users	34
5.3	Testing the Effect of Increasing the Starting Delay of Non-Milestone Users	36
5.4	Testing the Effect of Preloaded FSC Cache	37
5.5	Testing Bottleneck at the Database	38
Chapter 6	Validation	40
6.1	Model Validation	40
6.2	Statistical Validation	40
6.2.1	Number of Runs of the Code	41
6.2.2	Confidence Intervals	42
Chapter 7	Conclusions	44
Appendix A	Result Tables	45
References		49

List of Tables

4.1	Model parameters, values and descriptions for hardware elements	23
4.2	Model parameters, values and descriptions for software elements	25
4.3	File ID values for caching	25
4.4	Time distribution of users based on activity	28
4.5	Usage types and file sizes for PLM user roles	28
4.6	Model parameters, values and descriptions for user actions	29
A.1	Mean Response Time breakdown for FSC cache = 5 TB and all users starting together	45
A.2	Mean Response Time breakdown for FSC cache = 10 TB and all users starting together . . .	46
A.3	Percentage increase in MRT for Primary Designer (PD), Engineer (E), Manufacturing (M), Designer (D) and CAE Analyst (C) and varying number of Non-Milestone users	46
A.4	Mean Response Time breakdown for FSC cache = 5 TB and non-milestone users starting 15 mins late	47
A.5	Mean Response Time breakdown for FSC cache = 10 TB and non-milestone users starting 15 mins late	48

List of Figures

2.1	Model taxonomy for systems	7
2.2	Examples explaining (a) continuous and (b) discrete systems.	9
2.3	Steps in a simulation study	12
3.1	Network architecture for a single-user PLM Software	17
3.2	Network architecture for a multi-user PLM Software	18
4.1	Action Sequences for various PLM user activities	27
4.2	Flow diagram with simulation framework	32
5.1	Response time (in s) for various cache sizes and number of users	35
5.2	Cache Hit Ratio for various cache sizes and number of users	35
5.3	Response time (in s) for Primary Designer (PD), Engineer (E), Manufacturing (M), Designer (D) and CAE Analyst (C) and varying number of Non-Milestone users	36
5.4	Mean response time (in s) for various roles and Non-Milestone users having a longer start time delay	37
5.5	Impact of FSC cache size on Mean Response Time (in s) for Milestone users if the Non-Milestone users are initially delayed	37
5.6	Mean Response time (in s) for users with Preloaded Cache	38
5.7	Cache Hit Ratio for users with Preloaded Cache	38
5.8	Mean Response Time Trend for varying number of queries handled simultaneously at the database	39
5.9	Variation of Mean Response Time (in s) with number of queries handled simultaneously at the database	39
6.1	Number of queries for different operations at various times of the simulation	41
6.2	Mean Response Time (in s) averaged over completed simulations versus the number of completed simulations	41
6.3	Mean Response Time (in s) and its 95% confidence interval for the case with 1 query handled by the Database	42
6.4	Mean Response Time (in s) and its 95% confidence interval for the case with 10 queries handled by the Database	42

List of Abbreviations

Abbreviation	Description
PLM	Product Lifecycle Management
DES	Discrete Event Simulation
IT	Information Technology
CAD	Computer Aided Design
cPDM	Collaborative Product Development Management
QoS	Quality of Service
SLA	Service Level Agreements
FPGA	Field Programmable Gate Array
GWP	Google Wide Profiling
SimPy	S imulation in P ython
GUI	Graphical User Interface
RDBMS	Relational Database Management System
ClientHost	Client Host Computer
AppServer	Application Server
LFC	Local File Cache
FSC	File Server Cache
FCFS	First Come First Serve
LRU	Least Recently Used
D & R	Design and Release
CAE	Computer Aided Engineering
CN	Change Notice
MRT	Mean Response Time
CHR	Cache Hit Ratio

Chapter 1

Introduction

In the modern era, transportation and communication technology has led to the transformation of the world into a global village which is highly interconnected and interdependent [1]. A major aspect of globalization has been the growth of multinational corporations that have operations and distribution spanning multiple countries. To ensure the smooth functioning of these organizations, complex network infrastructure has been developed, which allows users in different centers of the organization, around the world, to continuously create, view and manipulate data. Not only does the data infrastructure and supporting software for these applications ensure secure and controlled access to essential files, but it also provides functionality to integrate the production activities in companies to business processes. The performance, reliability and availability of this data infrastructure thus forms the backbone for the proper functioning of such an organization and understanding and optimizing this network infrastructure can provide a company an edge over its competitors. Typically, in large scale production based companies, the above is expected to be achieved through the vision of Product Lifecycle Management (PLM) .

1.1 Product Lifecycle Management

1.1.1 Definition

PLM is defined as the process of managing the entire lifecycle of a product, from its conception, through design and manufacture, to service and disposal. A more comprehensive definition of PLM given by CIMdata, Inc. defines PLM as [2]:

1. A strategic business approach that applies a consistent set of business solutions that support the collaborative creation, management, dissemination, and use of product definition information.
2. Supporting the extended enterprise (customers, design and supply partners, etc.).
3. Spanning from concept to end of life of a product or plant.
4. Integrating people, processes, business systems, and information.

There are three major concepts in PLM [3]. Firstly, the product information should be universally available but selectively accessible. Secondly, all product information needs to be accurate throughout the lifecycle, including the information received from the extended enterprise of the products, namely the suppliers, vendors and customers. Thirdly, the PLM solution would also have to manage and maintain the business processes.

While PLM itself is not simply an IT solution, however, with the global nature of present manufacturing industry, the heavy dependence on technology and IT for most manufacturing processes and the large quantities of data and information that each company must maintain, the IT aspect of PLM has become very important. Thus, most large organizations require sophisticated and holistic software tools to ensure proper management of their products [4], as these companies make numerous products, with each product having hundreds, even thousands of parts and numerous engineers working on each part and product. Moreover, these software tools even integrate the extended enterprise into the system. Extended enterprise refers to the bodies that interact with the core company and includes the customers, service and logistic centers, suppliers and strategic partners [3].

1.1.2 Functions of PLM

The PLM software are designed to perform a number of major functions [3]. Firstly, they must allow people to create, share and collaborate. The data that can be shared includes CAD data, bill of materials, simulation studies etc. The second function they perform is that of Project Management. This includes keeping a track of project deadlines and milestones. The third function performed is Product Management. This function allows the management of the company to track the progress of the product in the market and alter the marketing strategy of the product if required. The fourth function performed is Portfolio Management. This is the higher order function which keeps track of product mix and profitability.

1.1.3 PLM Software

To provide the aforementioned services, a number of commercial PLM software solutions have been introduced by various companies, such as Teamcenter by Siemens [5] and Agile PLM by Oracle [6]. These software provide a variety of functionality such as CAD model modification, bill of materials management, lifecycle visualization, engineering data management, cPDM, supplier integration, content and document management etc.

The present research simulates the computer network of a PLM software using DES. It focuses on simulating those periods of operation when the company applies the greatest stress on the software's network, namely during product development deadlines, which is also when the company requires greatest productivity. The simulation study explores a number of vital applications. It simulates numerous *What-if* scenarios and helps carry out performance estimation of the network. It also helps fine tune the hardware and software parameters in the network and perform bottleneck analyses of various components of the network.

1.2 Motivation

As PLM software provide such a wide array of services, their networks are designed to offer massive data storage and speedy data access. Thus if the digitized form of all important files are to be stored and easily accessible by employees at the firm, the PLM network must ensure the following:

1. Data must be kept securely and be adequately backed up.
2. Data must be accessible quickly.
3. Due to the large amount of data to be stored, duplication of data must be minimized.
4. The PLM software must implement version control and access control.

Ensuring that the data is kept securely and the duplication of files is minimized has the important consequence that the firm providing the PLM software solution stores the data in a central server. However, this form of data storage implies that the data will not be accessible quickly. This problem is frequently seen in companies, where opening large files in the PLM software can take minutes. One of the measures that PLM software companies take to increase the speed of data access is to provide a large file server cache (FSC). This is a fast access data storage system which stores the most recently accessed files and thus reduces the time for data access.

While these delays are an inconvenience at most times, they are especially problematic during milestone events in the company. A milestone event is an event which covers the critical final period of a project where the design of the project may be frozen or some deliverables of the project may be due at the approaching deadline. During milestone events, the software users would like to function efficiently, with minimal delay. However, data access delay results in a hindrance in proper functioning. Thus there exists a need to analyze the functioning of such PLM software.

1.3 Related Work

DES is only one of the approaches that may be used to analyze computer networks. Other techniques such as analytic modeling, profiling and other simulation based techniques are also well established in literature for network analysis.

1.3.1 Analytic Approaches

There are numerous analytic approaches in literature. These approaches use different mathematical models to simplify the representation of networks, or individual elements of networks, such as application servers and database servers. Libman and Orda apply a game theoretic approach to model non-cooperative users sharing a resource over a network [7]. Tatawi et al. have used Markov decision processes to optimize resource allocation in multi-class networks [8]. Abdelzaher et al. have focused on the improving the quality of service (QoS) of web servers by applying classical feedback control to a control system representation of the World Wide Web [9]. A statistical model has been developed by Ahmad et al. to show that query interactions in the database can be used to improve database response [10]. Finally, there are a number of queuing models that have been used to represent network elements such as the application server [11], web server [12] and even multi-tier networks [13]. Urgaonkar et al. have also extended the multi-tier networks to incorporate provisioning algorithms that can dynamically reallocate resources to reduce overhead [14].

1.3.2 Experimental Approaches

Experimental methods involve setting up test beds to represent scaled down versions of the networks being studied and physically carrying out tests on this infrastructure. McWherter et al. show via experiments that the ability of the database tier to meet SLAs can be improved by using prioritized scheduling of queries [15]. Ellithorpe et al. have developed a test bed by using Field Programmable Gate Arrays (FPGAs) to construct hardware elements and showed that their setup could represent $O(10,000)$ node data centers [16]. Soundaryarajan and Amza showed that database tier response could be improved by using a K-nearest-neighbors machine learning approach to decide when to provide additional virtual resources to the database [17].

Profiling is an approach which samples data selectively from a functioning network and uses the sampled data to carry out performance analysis of the network. For accurate results, a profiling code needs to have sufficient sampling frequency but also very little overhead. Tallent et al. have developed a tool which uses profiling to study the bottlenecks in parallelization of code for petascale systems [18]. Ren et al. have

developed a tool that can be used to analyze data centers by collecting a variety of data, called Google Wide Profiling (GWP) [19].

1.3.3 Simulation Approaches

There are also numerous research papers which have used simulation based approaches to represent software networks and carry out performance studies, bottleneck analyses and even energy audits. Herrero-Lopez et al. have carried out simulations to represent a global infrastructure consisting of data centers with multiple tiers and have simulated the running of multiple applications on this network [20]. Lim et al. have developed a similar tool, MDCSim, which can be used to simulate multi-tier data centers [21]. CloudSim is another tool, which has been developed by Buyya et al. to represent virtual infrastructure based networks and to study their performance [22].

1.3.4 Reasons for using Simulation

There are numerous reasons for using simulation for the present study, as compared to the various methods discussed above. Firstly, the scale of the problem is very large and the problem is very complex. There may be as many as 500-1000 people working on the PLM software at once in a firm, the types of queries that pass through this system are different and there is a degree of randomness in the problem that makes finding an analytical solution to such a problem almost impossible.

Secondly, it is not possible for companies to set up experimental test beds for trying the various business and technology approaches. Such experimentation would require stopping the work of the entire firm or a portion of it, which in today's competitive manufacturing industry, is not something that the company can afford. Since global organizations never sleep, profiling approaches also become unfeasible, as the network is always functioning. Moreover, there are so many possible scenarios that physically testing them in a reasonable time frame would not be feasible. If a company conducts large scale exercises like these, it would be an immense financial strain to them.

All these reasons point to the setting up of a simulation study, which would be relatively inexpensive, would allow multiple scenarios to be run and would provide the results of the different scenarios quickly.

1.4 Research Contributions

The present research models a PLM software data storage network, representing its hardware, software and user attributes, with a wide range of parameters for each of these elements. It subsequently simulates the

network for numerous *What-if* scenarios. I feel that the following are the research contributions of this study:

- There have been numerous prior studies that have simulated data centers [20, 16, 21]. Herrero-Lopez et al. [20] have also simulated three different applications in their tool using message trees to represent the operations of each application. However, none of the prior art shows the modeling of actual business applications, with the variety of users and users actions, as has been done in this study. This study breaks down milestone operations into numerous users and the users are coded to deterministically perform operations, as they would in a real experimental scenario, thus allowing the study to be more accurate in terms of the results. Moreover the end objective of the present research is different from prior studies as it aims to improve the mean response time of milestone users, and have run scenarios that sacrifice the performance of non-milestone users to increase milestone user productivity.
- The present study focuses on improving PLM software performance. In the study of existing literature, while we observed numerous websites offering “tips and tricks” to speed up PLM software, we believe this is the first in-depth simulation based study of the software. However, it must be pointed out that this study has outlined the general approach to the simulation process. The present research extends beyond simply a PLM software as it shows the methodology that can be used to study any network, where there is an interaction between business process approaches and network software and hardware.

Chapter 2

DES – Theory and Implementation

There are numerous simulation approaches, which can be used to model a variety of problems depending on the type of system being modeled and the form of assumptions we are willing to make. A simulation is an approach in which a model is developed that imitates a real system. The model which is developed is based on a set of assumptions, where these assumptions introduce mathematical, logical and symbolic relationships between system entities [23]. A model taxonomy of the various types of systems that can be simulated, given by Fishman [24] is shown in fig. 2.1. There are certain characteristics, that make DES a very suitable technique to use for modeling the data flow through a PLM software network. To understand the same, one needs to understand the nature of discrete systems versus continuous systems. The subsequent sections define DES, explain the algorithmic paradigms that one can use to code a DES problem, the framework to approach and analyze such a problem and how it was implemented in the present study using a software called **Simulation in Python** (SimPy) [25].

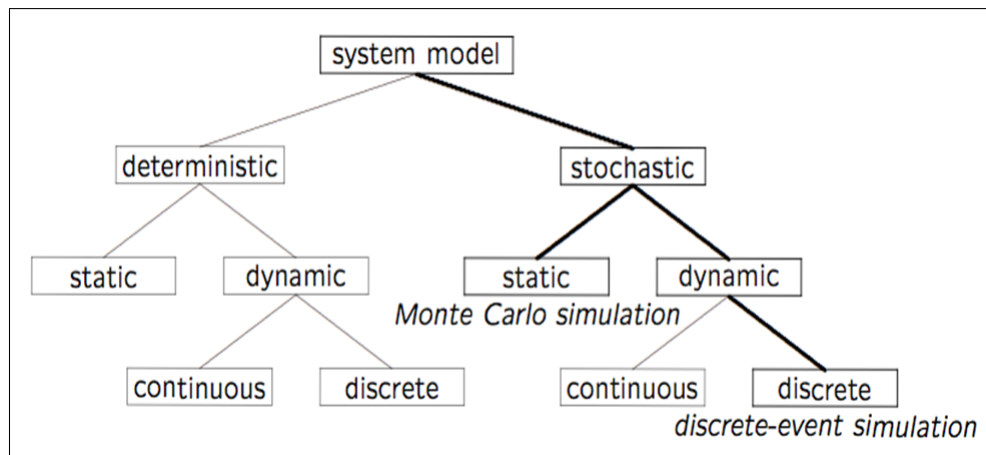


Figure 2.1: Model taxonomy for systems

2.1 Defining DES

Discrete Event Simulation can be defined as a modeling approach which studies events occurring at discrete intervals of simulation time using an event clock that progresses at discrete intervals. To understand this better, we need to understand the terms comprising the DES phrase, namely 'Discrete', 'Event' and 'Simulation'. The term 'Simulation' has been explained in the section above. The sub-section 2.1.1 explains the term 'Discrete' while the sub-section 2.1.2 explains the term 'Event'.

2.1.1 Continuous and Discrete Systems

In reality, any time increments that take place in a system occur in continuous time. In other words, the progression of time in a system that we are studying is continuous. However, in such a system, it may be possible that the events taking place in this system may be occurring continuously, or at discrete time intervals.

Consider two different problems. In the first problem, we wish to study the change in height in a bucket as water flows into the bucket at a variable rate. The change in height of the water in the bucket is a continuous variable and this system variable needs to be studied continuously to know its value as shown in fig. 2.2(a). Such a system, where the system variables being studied vary continuously are called *Continuous Systems*. On the other hand, consider a service center, where customers keep coming at random times and stand in a waiting line. The length of this queue is a quantity that varies only at discrete intervals of time as shown in fig. 2.2(b). Such a system, where the system variables being studied vary only at discrete intervals are called *Discrete Systems*.

It must be noted at this point that a system need not be discrete for using DES as an approach to study it. If a series of assumptions can be made such that a continuous system has system variables that are of interest, that vary only at discrete intervals, such a continuous system can also be simulated using DES.

2.1.2 Systems and System Events

A *system* is a group of objects that are functioning together to achieve a common purpose [23], such as a system of bank tellers, bank manager and bank resources together constitute a banking system. A *system environment* can be defined once we define a *system boundary*. For the banking system, money or customers entering the banking facility constitute the variables external to the system environment, and are hence outside the system boundary. An *entity* of a system is any object that is of interest to the study, and these are the variables that are kept track of in the study. The properties of these entities are called *attributes*,

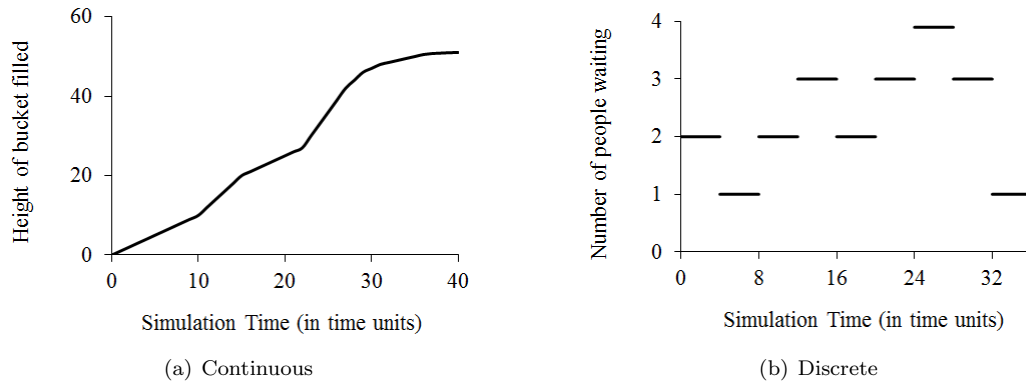


Figure 2.2: Examples explaining (a) continuous and (b) discrete systems.

which help define the system. An *activity* is a function that may be studied in the system, that may consume a certain amount of simulation time.

State variables in a system are those variables, which can occupy certain discrete states. Thus each combination of the state variable values together constitutes the *state* of a system. An *event* is any occurrence in a system that can cause the state of the system to get modified. To continue with the banking example, the state variables could be the state of the teller, whether he/she is busy or idle, the number of people waiting in the queue etc. So if there are two people waiting in the queue and the teller is busy serving a person, this is a state of the system. If another customer enters the bank and joins the line, or the teller is done serving the customer and becomes idle, this results in a change in the state variables and is known as an event.

2.2 DES Model Paradigms

Matloff [26] describes a number of simulation paradigms for DES. These can be described as approaches to implement DES by writing computer algorithms. He explicitly describes three models, the activity-oriented paradigm, the event-oriented paradigm and the process-oriented paradigm.

2.2.1 Activity-Oriented Paradigm

The approach to this paradigm is as follows. We initialize the model parameters and then discretize the simulation time period into tiny increments. At each time instant, we check if any event has occurred in the system, and modify the state variables if any change has occurred. In case of no change in the system, we simply increment the time step. Such a paradigm will only be suitable when the system being modeled is fairly continuous, and is generally not suitable for DES type problems, when the time between subsequent

events may be large.

2.2.2 Event-Oriented Paradigm

An important feature of DES is that although the events occurring in the system are stochastic in nature, to represent the nature of a real system, each event that occurs in turn initiates a subsequent event at a future time. Thus the event that is to occur right after an event occurs is always deterministic. This allows a paradigm where a priority queue keeps track of the next set of events that have to occur. A min-finding function can extract the earliest future time at which the next event will occur and will implement the change in system state variables corresponding to this change. A precaution in such an approach is that an event that occurs can make another event unfeasible or cause other events to occur. Thus an event manager is required to increment the simulation time at min-event increments and also delete or introduce events in the priority queue depending on the situation. This is the event-oriented paradigm.

While the event-oriented paradigm has lost some popularity over the years, and has largely been replaced by the process-oriented paradigm, it does have a number of advantages over other approaches. This approach is easier to implement, as the process oriented paradigm requires a thread-based implementation. The threads also slow down the implementation, and the event based approach is much faster. Lastly, the event based approach makes the code very flexible.

2.2.3 Process-Oriented Paradigm

The process-oriented paradigm is the one on the basis of which the code in this research project has been written. It is an approach which makes the DES algorithm flow more intuitively and makes the code easier to read and understand. Conceptually, it uses thread like structures to carry out DES. Each thread represents a different event. Thus in the case of the banking example, the arrival event would be a thread. Every transient entity in the DES system is assigned a thread and each permanent entity is assigned a queuing entity. Thus when a transient entity (such as a customer), which is modeled as a thread, wants to access a permanent resource (such as a teller), it enters this queuing entity and has to wait in line to access the services of the resource. The coding language that we use, called SimPy [25] uses this DES paradigm.

2.3 Framework of a DES Study

The DES model that is developed generally follows a framework, based on which the model is developed. A similar approach was followed in this study to develop the model. A breakdown of the steps of a simulation

study have been detailed by Banks et al. [23] and a flow diagram of the same is shown in fig. 2.3. The steps are as follows:

1. **Problem Formulation:** The first step in analyzing a problem is always formulating the problem statement.
2. **Setting of Objectives:** In this step the problem is further analyzed in a detailed fashion, to list down the objectives of the study and also decide that simulation is the best approach for the problem.
3. **Model Conceptualization:** In model conceptualization, the problem is finally studied in complete detail. The assumptions of the model are established, and the entities and relationships are identified. Subsequently, the model is refined till a final model of appropriate accuracy as compared to the real system can be obtained.
4. **Data Collection:** The previous steps develop a general model applicable in logic to the problem being studied. The data collection step converts the general model to one which exactly resembles the problem at hand. A fraction of the data collected can also be used later in model validation.
5. **Model Translation:** This step involves converting the existing theoretical model to a simulation code using either a DES software or by writing in-house code.
6. **Model Verification:** This step ensures that the simulation code performs based on the same set of relationships that were used to develop the theoretical model. This step generally involves a lot of debugging.
7. **Model Validation:** The validation step checks that the simulation results match with the results of the real system which was being simulated.
8. **Experimental Design:** Once the model which is being studied has been verified and validated, we can decide which other scenarios need to be simulated in the code.
9. **Production Runs and Analysis:** Once the experiments for the complete simulation study have been designed. They are simulated and the results of the simulation are collected. These results are then analyzed and relevant conclusions are made from the same.
10. **Additional Simulation:** Based on the results of the previous section, we decide if additional scenarios need to be run, and then proceed to run the same.
11. **Documentation and Reporting:** Based on the entire set of data collected, documentation of the model is done and a report of the results is prepared.

12. **Implementation:** If the recommendations of the simulation study suggest some changes in the current system, the same can be implemented and over time the results of the modification can be studied to see the results of the change. One may need to change model parameters and re-simulate, depending on the changes in the real system viz-à-viz the changes simulated in the simulation study.

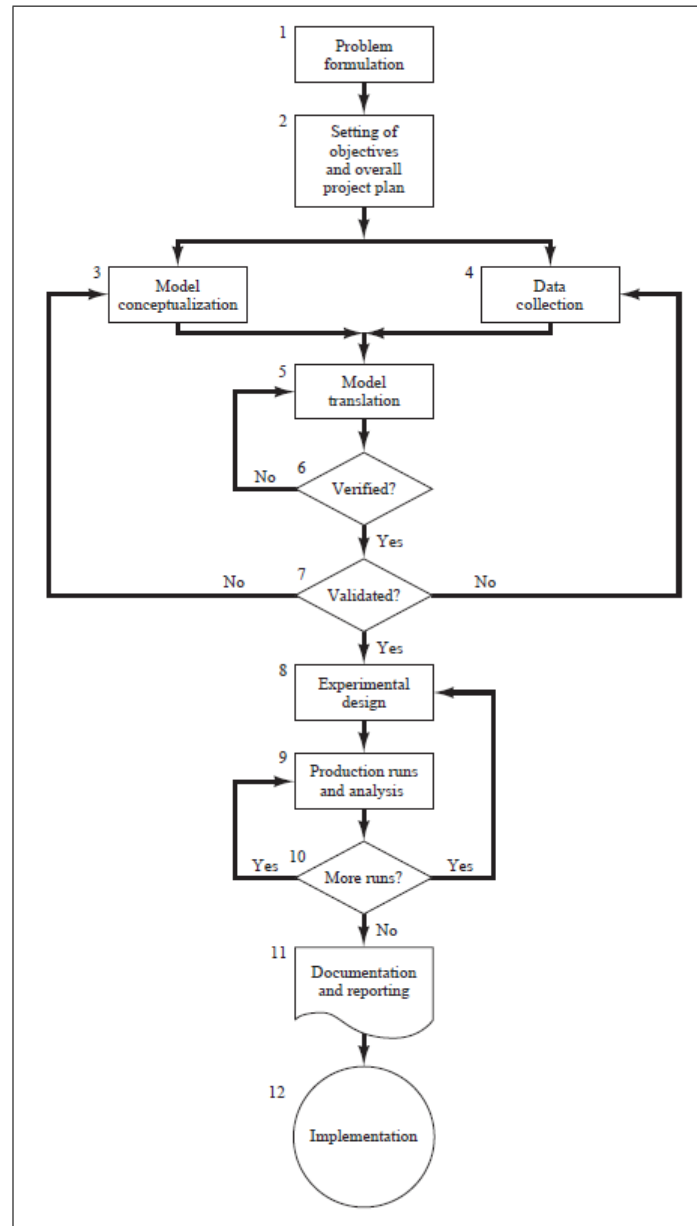


Figure 2.3: Steps in a simulation study

2.4 DES using SimPy

Due to the level of complexity the code to be written, the language required to code needs to be robust but at the same time very flexible to allow quick modification of the code to run different simulation scenarios. Moreover it needs to have the basic functionality of carrying out DES. A module named SimPy [25] (has been developed as an open source module by the “SimPy Developer Team”. It is an object-oriented, process based language written in Python and allows the coder to easily implement the basic features of DES. In this coding approach, the “customer”, or transient entity is represented by a function. The nodes or permanent entities are represented by classes called “Resource”. The SimPy module implements a form of multi-threading such that the transient entities represented by functions are each given a thread. The threads are executed in the order of time of execution i.e. the event stepping that is done in DES, and the event list that is maintained in DES is maintained internally by SimPy and this list stores and executes threads one by one. If a resource is occupied by one thread and another thread tries to access it, the second thread has to wait till the resource becomes free. The resource can also have multiple servers i.e. a resource can have a size, equivalent to a multi-server queue. Other features which have been provided in this module are the notion of priorities, preemption, balking and reneging.

Another very useful feature in SimPy is the collection of statistics. The module provides a feature called “Monitor”. Each monitor is of the form of a dynamic array, whose size can change as more data is collected. To store data in a monitor, a function *observe()* is passed the value to be stored and it stores the value in the monitor along with the Simulation Time at which the event occurred. These monitors can later be studied to collect the required statistics. There are a number of software packages similar to SimPy in many other programming languages. However, the reason I prefer Python and SimPy are the following advantages it offers:

1. Python has a robust random number generation module called “random” which makes it easy to generate random variates from all types of distributions.
2. Python is among the most recent coding languages, and the designers of Python have focused on making it a language which is very easy to code in and have numerous features which speed up the process of quick code development. The language, amongst other features, has an indentation based grouping of code which makes it easy to code and also has no fixed data types, which gives a great deal of flexibility to the programmer. Python also has a very large support base (help forums and tutorials) which means any problem you face can be solved.
3. The reason why SimPy is a good simulator is that it has a very dynamic set of developers who are

very active on the mailing group and are constantly developing and updating the source code. The source code is open source and hence gives the option of modification in the source code itself if the need arises. Lastly, there is a very good tutorial set on the SimPy website which makes the process of understanding how to code in this language very straight forward.

4. Since SimPy is written with the process-oriented paradigm, it makes writing the code more intuitive as compared to the event-oriented paradigm.
5. Another advantage of SimPy is the in-built features that allow a thorough step-by-step analysis of the DES system, to ensure its correctness, by using the Simulation Trace feature to step through the DES, Monitors to store data during the run, and SimGUI, to analyze the flow of transient elements in the system visually using the SimPy GUI.

Chapter 3

Network Topology of PLM Software

To understand the commercial state of the art, we studied the user manuals of the Siemens PLMs Teamcenter [5] solution suite; it is used as the typical PLM software at most automotive manufacturers. The models described in this thesis are based on our study of these manuals and are not intended to capture or represent any real implementation in any manufacturer.

To simulate the network of a PLM software, the network needs to be broken down into its constituents. This section discusses the elements composing the PLM network. In the subsequent section 4, a model is described for each individual part of this network and the implementation of each element in SimPy is also discussed. We study the network architecture first at the individual level of one user using the network and then at the system level, which discusses the interactions between ‘n’ users who are using the network simultaneously.

3.1 Single-User Network Architecture

As discussed in the introduction, a PLM software is not simply a tool on a single computer, but in fact an entire network architecture which ensures that a large amount of data is accessible to multiple users and is kept securely and at a high availability rate. Let us consider the actions of a user who uses this tool. The user initially logs into the software and the software authentication takes place. Then the user will probably load a file and start working on it. Once he has completed his work, he will save the results. In each of these steps, a number of operations are taking place in the background, to ensure that the user has a seamless experience. The login step requires a comparison of the login information of the user with the list of user names and passwords which would be stored somewhere in the server system. Then, the file which is to be worked upon, is initially located on the servers of the PLM software network, and has to be fetched. Modifications to the file are an operation which takes place locally on the user’s computer, but may still be taking place on the PLM software. The ‘save’ step once again involves transferring the file from the user’s computer to the secure data storage, which is a network operation.

3.1.1 Formal Description of the PLM Network Topology

The network topology of the PLM software simulated in the research has been shown in fig. 3.1. The different types of commands that are given by users at the PLM software interface, such as ‘launch’, ‘open’ and ‘save’, generate file access queries, and the flow of these queries through the system has been simulated in this study. The following steps result in the conversion of the query into a meaningful result which is displayed on the user’s display screen via the PLM software.

1. When the user of the PLM software or the ClientHost gives a command on the PLM software, a query corresponding to the command, is generated. The queries which originate at the ClientHost, first travel to servers called application servers or AppServers. An AppServer is a program, or in this case, a separate machine, that handles all the operations that link up the users to databases.
2. A Database is a network node that contains an installation of RDBMS tools. The Database interprets the queries that were sent from the ClientHost via the AppServer and deciphers the query to give the location/locations of the file/files the Client wants to access. The term ‘ClientHost’ simply refers to the users who have a dedicated PLM software installed on their workstation.
3. These messages are returned to the AppServer, where they are routed back to the ClientHosts. The ClientHost has two components, the Client, which is the dedicated PLM software on the ClientHost’s workstation, and the Local File Cache or LFC, which is a fast access storage on the workstation of the user. The LFC ensures that if a file is repeatedly accessed by the user, it is stored on his workstation and can be pulled up quickly.
4. If the file that is being sought is found in the LFC, the file is opened on the PLM software and the process is complete. On the other hand, if the file is not present in the LFC, then the query returns to the Client and is sent forward to the File-Server-Cache Server or FSC Server.
5. The FSC Server is designed to route queries from the ClientHost to the File Server Cache or FSC. In functionality, it is modeled as similar to the AppServer.
6. The FSC is a cache similar to the LFC, but has a much larger storage capacity as compared to the LFC, as it has to hold files being accessed by the numerous users of the PLM software. The FSC forms the second tier of file caches, and using these two tiers, the PLM software network tries to ensure speedy access to files.
7. If the file that is required is found in the FSC, the file is returned to the user, otherwise, it is fetched from the Volume.

8. The Volume is a large data warehouse, which archives the complete set of data of the company which is using the PLM software. The reason why there are two levels of file caches is that the file access requests that reach the Volume take a lot of time to obtain, due to the sheer size of the data stored in it.

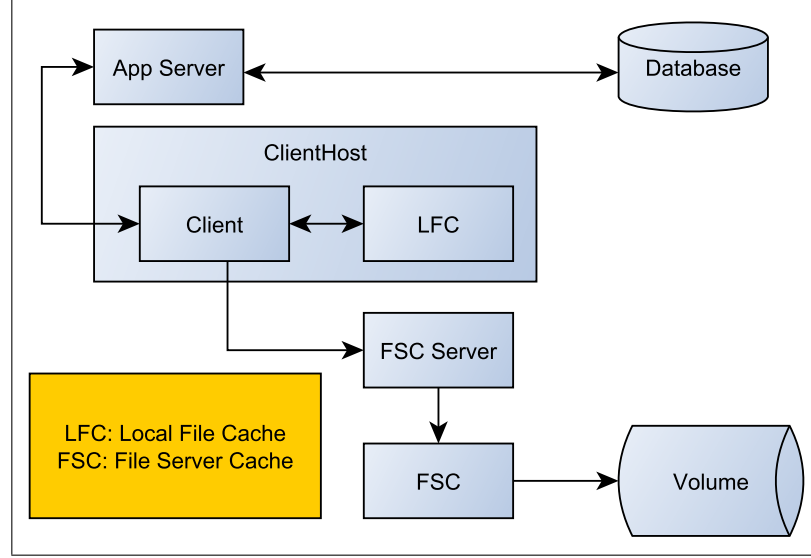


Figure 3.1: Network architecture for a single-user PLM Software

3.2 Multi-User Network Architecture

The Single-User Network Architecture considers that each user of the network is independent. However, the problem being addressed in this study relates to the fact that as many as five hundred users of the PLM software may be accessing the resources on the network simultaneously. It is in such cases, that we need to study the entirety of the model. Thus the complete model that has been studied in this research project. This network is very complex as it involves a high degree of interaction between the queries of different users. A flow diagram of this system is shown in fig. 3.2.

The components of this system are the same as those of the single-user architecture. However, the presence of multiple users introduces a number of fundamental differences in the analysis.

1. The users each have individual AppServers, which implies that these cannot be the sites at which bottlenecks will occur.
2. However, the Database, Volume, FSC and FSC Server are shared by all the users. Hence these can be potential sites for bottlenecks.

3. Since a number of queries may simultaneously access some of the resources, the model that is developed must implement these resources to have queues as the resource may not be able to handle all these queries simultaneously. This does not imply that these resources only handle single queries at a time. A number of the resources can handle multiple queries simultaneously, and in our research the Database and Volume are designed to possess this property.

At this stage, it becomes important to finally understand the complexity of the simulation being carried out. Each of the operations of the various users takes varying amounts of time. Moreover, the time taken to respond to the user's command depends on the size of the file being accessed, number of simultaneous users accessing files, the location of the network where the file is located and other reasons. The different users have been modeled to have different properties, and work at different rates. Section 4 will explain the implementation of components of this network, the breakdown of times at each step of the network and representation of the PLM software in the simulation code.

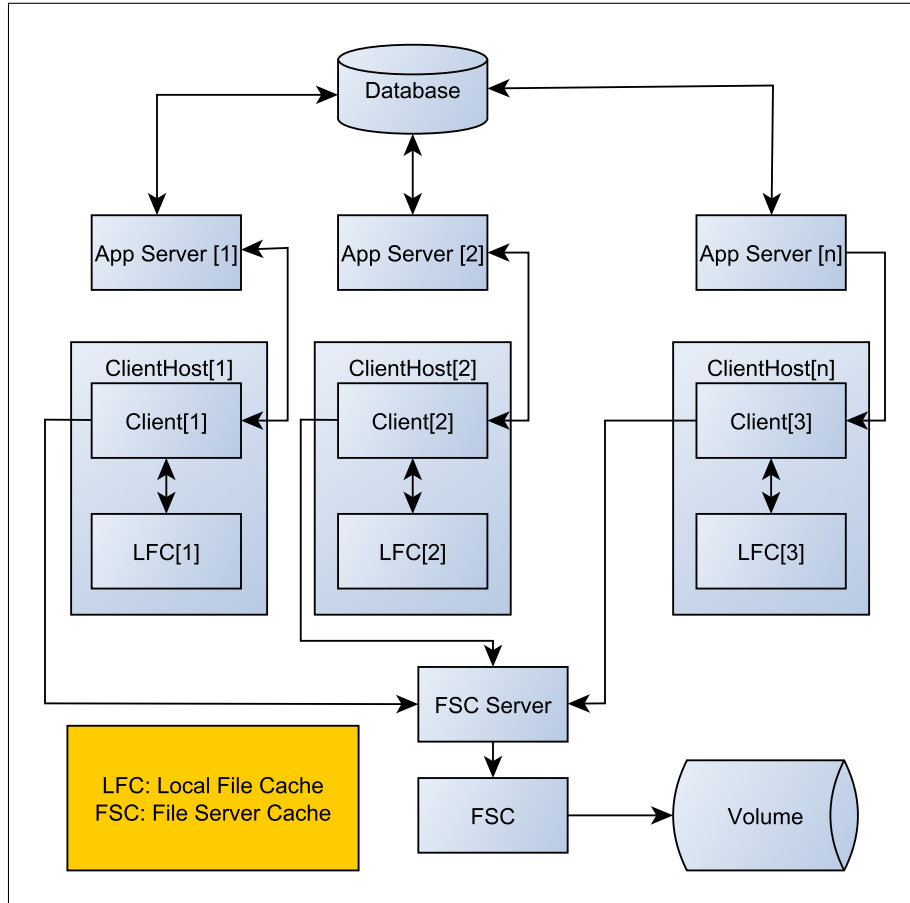


Figure 3.2: Network architecture for a multi-user PLM Software

Chapter 4

Modeling Methodology

The very robust and user-friendly DES Tool, SimPy [25] was used for developing the simulation model. The tool has inbuilt simulation time management using an event dispatcher and a process based approach which greatly simplified the task of building the model. However, the tool is a generic DES modeling tool and modeling both the hardware and software aspects of a multi-user PLM software used various approaches, which are explained in this section.

It should be noted that the models were derived from our study of an initial internal report by BIMCON Inc. [27] that was made available to the researchers so that we could understand and model the complexities of the business process and hardware that is typical of a commercial PLM software.

There are three tiers in the approach used to model the PLM software network. The first is hardware modeling, which is the representation of the hardware components of the network in the simulation code, the second is the representation of the software components of the PLM software, for accurately replicating the software's functionality and the third is the approach used to represent the business functionality of the tool which forms the interface between the first two steps. Each of these elements has a number of parameters associated with them, which allows parametric studies on the PLM model. In this chapter of the report and subsequent chapters, the term “milestone” will be frequently used. This section begins by giving a definition of the term and then describes the three tiers of the modeling methodology. Finally the framework of the entire simulation code is explained.

4.1 Concept of Milestone

A Milestone event is an event which covers the critical final period of a project where the design of the project may be frozen or some deliverables of the project may be due at the approaching deadline. They are a useful business methodology as they help in accurately determining if the project being worked upon is being completed on schedule. In the results chapter, chapter 5, we assume that a team of the PLM software users are trying to complete a milestone.

4.2 Hardware Modeling

The hardware components of the PLM model include the transient components, namely queries which move between the permanent components or hardware resources such as the Clients, Servers, Caches, the Database and the Volume. Queuing models have used to represent the permanent components of the model, with queries enqueueing and dequeuing from the components. The queries originate at the Clients and proceed to follow the path as described in chapter 3.

1. In SimPy, the queries are represented as objects of the process class. Each user of the PLM software is assigned a query, whose parameters gets modified once a particular query is completed. Hence a query is equivalent to a user. The process class is where the functionality of the queries are written. Hence the syntax of the process class is as follows:

Listing 4.1: Process class syntax

```
1 class PLMMessage(Process):
2     ...
3     def __init__(self, name):
4         <set object parameters here>
5         ...
6     def runMsg(self):
7         <write the run code for a query here>
8         ...
```

Similarly, once the process class is written, the syntax for the queries is to make objects of the class and activate the threads for each query using the *activate()* command. The code snippet for this task is given below:

Listing 4.2: Query syntax

```
1 def main():
2     ...
3     for i in range(numClients):
4         name = 'Msg%02d' %(i)
5         PLMMsg = PLMMessage(name)
```

```

6      initDelayTime = G.Rnd.uniform(DelayParLow, DelayParHigh)
7      activate(PLMMsg, PLMMsg.runMsg(), at=initDelayTime)
8      simulate(until = maxSimTime)

```

In this code, for each Client (total number of clients equals numClients), the query gets a name, an initial time delay is chosen and the query is activated. The total simulation time equals maxSimTime, a parameter which the programmer can set.

2. Using Kendall's notation [28], FSC server and FSC caches have been represented by M/M/1 FCFS queues. As per the assumptions we have made for the model, the data access time for the caches is negligible as compared to the time taken in the database and the volume. This allows us to model them as M/M/1 queues as the queries accessing these resources are serviced instantaneously. These permanent entities are coded as objects of the resource class:

Listing 4.3: Resource syntax for FSC Server and FSC

```

1  FSCServer = Resource(1, name = 'FSCServer')
2  FSC = Resource(1, name = 'FSC')

```

3. The Clients and AppServers and the LFC are all resources that are always available. Hence they have been modeled as M/M/ k_1 FCFS queues, where k_1 is the total number of users that are using the software. These entities have been coded as:

Listing 4.4: Resource syntax for Client, AppServer and LFC

```

1  client = Resource(numClients, name = 'Client')
2  appServer = Resource(numClients, name = 'AppServer')
3  LFC = Resource(numClients, name = 'LFC')

```

4. The Database and Volume are modeled as M/M/ k_2 FCFS queues, where k_2 signifies the number of queries the Database and Volume can handle simultaneously. These entities have been coded as:

Listing 4.5: Resource Syntax for Database and Volume

```

1  database = Resource(databaseMsgs, name = 'Database')
2  volume = Resource(volumeMsgs, name = 'Volume')

```

5. The LFC and FSC caches file storage mechanisms have been simulated as LRU caches. The caches are modeled as objects of a Cache class, which contains the functionality of the caches. The Cache class takes as input the LFCCacheCapacity and FSCCacheCapacity, which are the sizes of the caches.

Listing 4.6: Cache object syntax

```

1 LFCCache = []
2 for i in range(numClients):
3     LFCCacheObj = Cache(LFCCacheCapacity)
4     LFCCache.append(LFCCacheObj)
5 FSCCache = Cache(FSCCacheCapacity)

```

6. The queries are generated based on Poisson arrivals rates. Once the queries are moving within the simulation system, the delays that the queries experience are uniformly distributed with different ranges of values. In some cases the delay is dependent on the size of the files. While the files move from one hardware resource to another the rate of file transfer has been modeled as a function of the data transfer rate (in Mbps).

The simulation model has been developed to be very flexible, so that a number of parameters in the system can be varied, to carry out parametric analysis on the model. Some of the important hardware parameters are cache sizes, various file delays, data access rates and the queries handled simultaneously by the Database and the Volume. These parameters have been further enumerated in table 4.1.

4.2.1 LFC and FSC as LRU Caches

LRU caches keep track of the cache elements in the order in which they were used, and first discard the cache items which have been the least recently used. Thus in the current implementation, they have been represented by priority queues ordered on the arrival time stamps of files. The maximum size of the queue is the cache size. When a new cache element is to be entered into the queue, if there is sufficient space in the cache, the new element can simply be added. If the cache does not have sufficient space, instead of a single message getting removed from the queue, the least recently accessed messages are removed, based on least priority of time stamp till the cache has enough space to accommodate the new item.

Cache hits can significantly reduce the file access time, as they reduce the network length to be traversed by the files to reach the Clients and also save unnecessary volume accesses. A parameter that was monitored

Table 4.1: Model parameters, values and descriptions for hardware elements

Parameter	Value	Description
FSC cache size	0.5-10 TB	Amount of storage in the FSC
LFC cache size	100 GB	Amount of storage in the LFC
Queries handled by Database	1-10 nos.	Number of queries that the Database can handle simultaneously
Queries handled by Volume	1-10 nos.	Number of queries that the Volume can handle simultaneously
Volume data access rate	800 Mbps	Speed at which the Volume transmits data
LFC data access rate	800 Mbps	Speed at which the LFC cache transmits data
FSC data access rate	800 Mbps	Speed at which the FSC cache transmits data

in this study is cache hit ratio (CHR), defined in eqn. 4.2.1.

$$\text{Cache Hit Ratio} = \frac{\# \text{ of cache finds}}{\text{Total } \# \text{ of cache accesses}} \quad (4.2.1)$$

Listing 4.7: Cache class syntax

```

1 class Cache:
2     def __init__(self, sizeofCache):
3         # cache takes elements of the following format:
4         # cache element = [msgID, dataSize, timeStamp]
5         <Cache Parameters here>
6         ...
7     def isMessageInCache(self, msgPointer, currentSimTime):
8         <Function to check if a message(msgPointer) is in cache>
9         ...
10    def writeMsgToCache(self, msgPointer, currentSimTime):
11        <Function to a message(msgPointer) into the cache>
12        ...

```

4.3 Software Modeling

Section 4.2 details the modeling approach used to represent the hardware of a PLM software network as a DES model. This hardware functionality is hand in hand with the software functionality of the PLM software. This section describes the software modeling approaches used in this study.

A PLM software functions on the basis of user events. These are user initiated actions on the PLM interface. Examples of this include ‘load’, ‘save’ and ‘search’. The type of operation being carried out on the software affects the response that the software has. For example, a load command can find the file being sought in the LFC, the FSC or the Volume, depending on which of the tiers it was found to be available first. However, a save command visits all three of these tiers and saves a version of the file in each of them.

Based on industry research, we decided to model product design software usage scenarios in our research. We established that product designer teams working on a design freeze milestone on a PLM software perform certain operations, which we refer to as Launch, Search, Load, Zoom, Save, Review, Snapshot, Workflow, Notify, Geometry, CN and NonPLM. These operations converge to the following operations based on the flow of information in the PLM network:

- **Launch:** Action of launching the PLM software.
- **Search:** Action of searching for a file or multiple files.
- **Read:** Action of requesting a particular file e.g. Load.
- **Local:** Actions using local features of PLM Software e.g. Zoom, Review and Snapshot.
- **NonPLM:** Actions which do involve the PLM Software e.g. Geometry.
- **Write:** Action of saving a file or overwriting it e.g. Save.
- **Workflow:** Actions that assist in workflow monitoring e.g. CN, Verify and Check.
- **Notify:** Action of notifying stakeholders.

The simulation code takes into account the type of query in the network and the model adjusts its behavior according to the type of query. The type of operation also impacts the size of the files being accessed. Launch and Notify files are very small in size, while read and write files may be much larger. Finally, the different types of actions are assigned different types of file names. The number of file names is finite to develop a realistic scenario. For example, milestone user data files were assigned file names from 1-1000 to represent the fact that milestone users refer to a finite number of files during the final stages of a project. These parameters have been further enumerated in tables 4.2 and 4.3.

Table 4.2: Model parameters, values and descriptions for software elements

Parameter	Value	Description
Database retrieval delay	(1,10) s	Range of times of delay to run a query through a database
Large file size	(900,2000) MB	The range of sizes that "Large" files can have
Medium file size	(500,899) MB	The range of sizes that "Medium" files can have
Small file size	(100,499) MB	The range of sizes that "Small" files can have
Tiny file size	(10,50) MB	The range of sizes that "Tiny" files can have

Table 4.3: File ID values for caching

File Category	ID nos.
Milestone Client Files	(1,1000) nos.
Other Client Files	(5000,8000) nos.
Milestone Launch Files	(10000,12999) nos.
Other Launch Files	(13000,16000) nos.
Workflow Files	(20000,25000) nos.
Notify Files	(30000,31000) nos.

4.4 Business Process Modeling

In any network, there are three interacting entities, the hardware, software and users. In a PLM software, accurately representing the actions and behavior of business users is of prime importance. This section explains this enabling layer, that is responsible for the interaction of hardware and software.

As explained earlier, the simulations conducted in this study relate to the modeling of a company in which a team is working to complete the work for an approaching milestone, and the milestone operations are being studied. This team of users performs the following three major activities: Design checking, Issue Resolution and Design Release. The users have various roles, such as Primary Designer, Engineer, Designer, Manufacturing Engineer and CAE Analyst, who spend varying amounts of times on the each activity. The following are the assumptions that have been made in implementing the concept of simulating a milestone

event:

1. **Business event/Milestone period:** 10 days; 8 hours per day. This is based on the final 2 weeks prior to the milestone date.
2. **Number of sequences executed during the milestone:** 200 (Assumption: 1000 checks average during the design phase of which 80% completed before the milestone period. This information is useful to project the typical number of sequences each member will complete in a day).
3. **Scenario period:** One day (8 hours) of the above mentioned period.
4. **Activities undertaken:** Design Checking, Issue Resolution, Design Release
5. **Teams:**
 - Design checking: 10 teams
 - Packaging Teams: 4
 - Design Engineering Teams: 3
 - Manufacturing Teams: 3
 - Issue resolution: 5 teams Vehicle Attributes Teams (including CAE): 5
 - Design release: 10 teams
6. **Member load distribution across the three activities:**
 - (a) We assume that 100 people are working on the milestone, with the following distribution:
 - 38 Designers
 - 7 Primary Designers
 - 7 D & R Engineers
 - 21 Manufacturing Engineers & Designers
 - 27 CAE Analysts
 - (b) At various times in a day, the distributions of these users will vary based on their roles, across the teams and the activities. We have assumed a representative day with the allocation of the different members to the three activities (summing up to 100% of the time, i.e., 8 hrs.) shown in table 4.4.
7. Depending on the role of the user, the files they access may be of various sizes. The typical event sequences for each of the three activities is given in fig. 4.1.

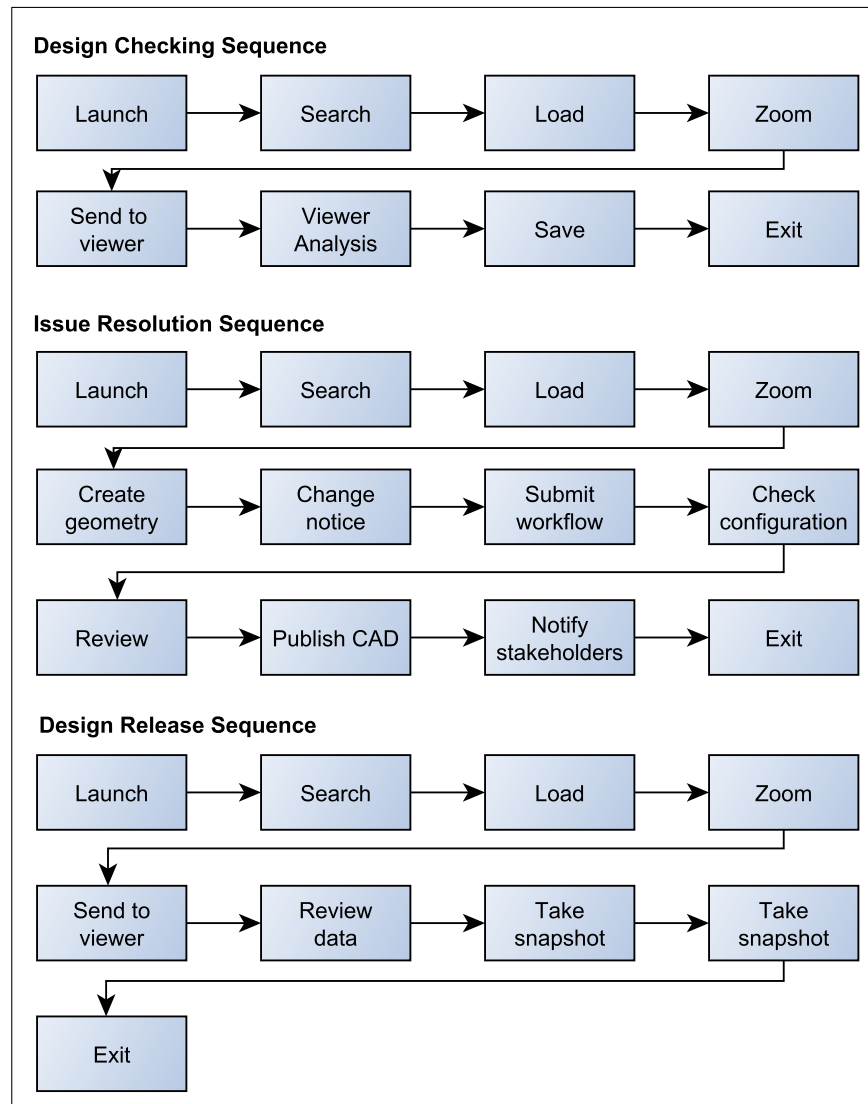


Figure 4.1: Action Sequences for various PLM user activities

Table 4.4: Time distribution of users based on activity

Member Role	Nos. in Design	Nos. in Issue	Nos. in Design
	Checking	Resolution	Release
Primary Designer	40%	20%	40%
Engineer	33%	33%	33%
Designer	33%	33%	33%
Manufacturing	33%	33%	33%
CAE Analyst	0%	100%	0%

8. Table 4.5 gives a break down of the number of times different users carry out these activities during a day and the sizes of the files that are being moved during these action sequences.

Table 4.5: Usage types and file sizes for PLM user roles

Member Role	Usage Type	File Size for each sequence (MB)
Designer	Heavy (>10 sequences per day)	Medium (500-900)
Primary Designer	Intermediate (3-10 sequences per day)	Small (<500)
Engineer	Intermediate (3-10 sequences per day)	Small (<500)
Manufacturing	Heavy (>10 sequences per day)	Medium (500-900)
CAE Analyst	Light (<3 sequences per day)	900-2000

9. Another aspect of the simulation tries to replicate the fact that apart from milestone users, there are other non-milestone users who simultaneously use the PLM software with the milestone users. Since the immediacy of their work is different, the simulation model that has been developed allows the variation of the number of non-milestone users and parameters for milestone and non-milestone clients can be modified independent each other. These parameters include the start time of the non-milestone users, the kinds of actions they carry out and the amount of break they take between sequences. These

parameters have been further enumerated in table 4.6.

Table 4.6: Model parameters, values and descriptions for user actions

Parameter	Value	Description
Number of users	100-500 nos. (Milestone users: 100)	This is the total number of people using Teamcenter at a company, who share a data volume
Simulation Time	28800 s	Total time of operation in a day being simulated
Local operation delay	(300,600) s	Range of times of delay when a user is working on a local operation (e.g. Zoom). During this time the user is busy, but no queries are being sent through the network.
Non-PLM operation delay	(600,900) s	Range of times of delay when a user is not working on a PLM software operation (e.g. using CAD software).
Heavy user inter sequence delay	Exponential ($\lambda = 120s$)	Range of times of delay between running two sequences for Heavy Users”.
Moderate user inter sequence delay	Exponential ($\lambda = 900s$)	Range of times of delay between running two sequences for Moderate Users.
Light user inter sequence delay	Exponential ($\lambda = 21600s$)	Range of times of delay between running two sequences for Light Users.
Non-Milestone Multiplier	2x	The delay between two Non-Milestone user queries is twice that of the Milestone users.
Milestone user starting delay	(1,900) s	Range of times of delay between the start time of various Milestone users.
Non-milestone user starting delay	(1,2700) s	Range of times of delay between the start time of various Non-milestone users.

4.5 Simulation Framework

The sections 4.2, 4.3 and 4.4 give implementation details of the hardware, software and business modeling tiers of the code. In this section, the other elements of the SimPy code which help carry out a statistically significant and valid simulation, and elements that store data for analyzing the system are discussed and a framework for the code has been described. A flow diagram of the framework is shown in fig. 4.2. The steps in the flow diagram are explained:

1. **Configuration File:** The Configuration file or Config file (in green in the flow chart) is a separate

file which is read into the main code (in blue in the flow chart). As the data being contained in the simulation code file can become overwhelming, an attempt was made to modularize the code and separate the inputs. To do the same, configuration files were developed which contained the inputs to the simulation file. Thus any modifications to the inputs could be made in a separate file and the simulation file would be generic and would work with different input values. A Python configuration file module called *ConfigParser* was used for the same. The module has a few useful features:

- It allows the input file to be sectioned, such that the same parameters can be given different values depending on which section is read. Thus if different simulations need to be conducted, the different parameters can be put in different sections.
- It creates a Config file which is properly aligned and simple to read, following a certain format. Thus the Config file can itself be modified without having to modify the Config-File generating source code.
- Another very important advantage of a Config File over simply storing this information in an input file, one needs to keep track of the line numbers of the data. Such a problem is not experienced in Config files, which follow the format:

Listing 4.8: Config file syntax

```
1 config.get (section-name, param-name) = param-value
```

2. **Setup Cartesian Product:** Once the Config file is entered into the code, it is read by the main file. To allow the possibility of entering multiple scenarios into a single simulation, the code allows the Config file to accept a range of values for the different parameters. These ranges of values are recognized in the input module of the code, and a list of simulation run parameters is setup. This list of runs is obtained by taking a cartesian product of the input variables i.e. take all possible combinations of the variables and creates a list, whose elements are the inputs to individual runs. This list is also advantageous if the code were to be parallelized in the future, as the simulation with each element of the list would be completely independent of the other, and could run on a different core.
3. **Set Parameter Values:** Once the cartesian product of the input variables has been taken, we have a list, whose elements are inputs to each simulation run. We start the simulation by taking the first run and setting the parameters of the model with the values from this list item.
4. **Set/Reset System Elements:** There are a number of system elements in the simulation code which need to be reset with each run. These include resources such as the Database, Volume, FSC Server,

AppServer and the Client, the caches and Monitors.

Monitors are in-built SimPy objects for storing time related data for statistical analysis. They allow for easy data collection and store the collected data in a simple format so that data can be easily extracted. Monitor data is collected during the run and needs to be post-processed to analyze the results of the run. The syntax for the storing a useful piece of data in a monitor is given by:

Listing 4.9: Monitor syntax

```
1 Monitor-name.observe (data-to-store)
```

The time information to store may be the waiting time, processing time or any other piece of data that may need to be stored. The monitor stores the data and adds a time stamp of the current simulation time to keep track of when the stored event occurred.

Another important aspect of a DES simulation is that to get statistically valid results, the simulation for each parameter set needs to be run multiple times. Thus at this stage a seed value is also set for the random number generator. For each new seed value also, the above system elements are reset.

5. **Setup Threads Representing Individual Users:** At this stage the simulation can be started. Therefore, the threads for the user queries are set up and initialized.
6. **Run Threads:** Once the thread initialization is complete, SimPy runs the simulation till the simulation time is complete.
7. **Store Monitor Values:** Once a run is complete, the data from the monitors is extracted, since the monitors will be reset. this data is then stored in a suitable form.

Now, if the total number of repetitions for the simulation run are not complete, a new seed value is set and a loop returns the system to the “Set/Reset system elements” stage. If the repetitions are complete, the system increments the simulation run number and returns to the “Set parameter values” stage for a new scenario run.

8. **Store Statistics:** Once all the scenarios have been run, the statistics for the scenarios based on the data stored from monitors is collected and stored. These statistics can then be used to draw inferences.

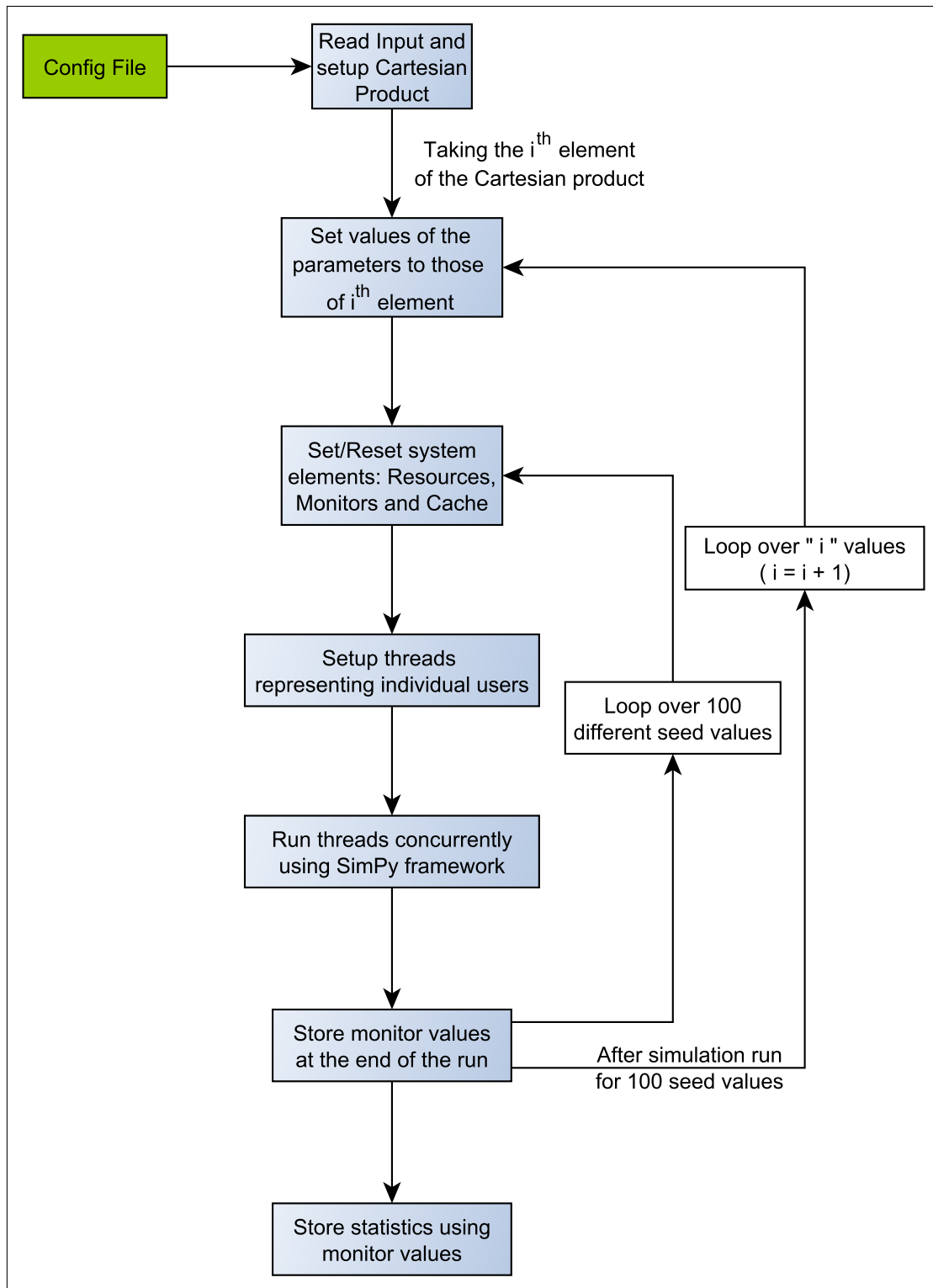


Figure 4.2: Flow diagram with simulation framework

Chapter 5

Results and Discussion

A number of *What-if* scenarios have been tested in the simulation study. It should be noted that these were derived from our studying of an initial internal report by BIMCON Inc. [27] that was made available to the researchers so that we could understand and model the complexities of the business process.

The aim of these scenarios is to try to increase the efficiency of operation and productivity of the milestone users of the PLM software. The variable which will be used to define this productivity is the Mean Response Time (MRT), which is defined as the average time required by a query which originates at the ClientHost to complete a tour of the network path, as described in section 3, namely from ClientHost to Database, from the Database to the caches or Volume, and back to the ClientHost.

Another variable that will be studied is the Cache Hit Ratio (CHR) for the FSC, as increasing this variable will reduce the path that files have to travel, and hence in most cases this should reduce MRT. For all of these scenarios, the number of milestone users and their actions are kept the same. The reason for this is that during milestone operations, the number of milestone users and their roles are pivotal to the completion of the project. Thus even if the other users may alter their activity, the milestone users must continue using the PLM software are necessary.

5.1 Testing the Effect of FSC Cache Size

The FSC cache size can be chosen by the firm when the PLM software network is initially being setup. Thus a simulation of different FSC cache sizes can allow the company to decide, based on the numbers of users and the workload they have, the size of FSC cache that they should choose.

Figs. 5.1 and 5.2 show the results from the simulation study carried out. The trend from the results shows that increasing the FSC cache size increases CHR and decreases the MRT. Such a result is consistent with expectation, as a larger FSC cache size implies that more of the files being sought by users in the system would be stored in the cache. Hence to access them, the time for data access would be less as the data would have a greater probability of being found in the cache, and therefore the query would not have to

go to the Volume. Moreover, since more of the files are being stored in the cache, CHR increases. However, the magnitude of the increase in CHR and decrease in MRT does not show a linear trend. We can observe that increasing the cache size from 1 TB to 10 TB decreases the MRT by only about 5 – 8% even though the CHR can increase by 4.5 times.

A counter-intuitive observation is seen for the case where the total number of users are 350 and 500. In these cases, for very large cache sizes, the CHRs continue increasing, but the MRT becomes unresponsive. This helps us conclude that the increase in CHR and decrease in MRT, though generally showing direct proportionality, can become uncorrelated for very large cache sizes.

A possible explanation for this trend could be that for very large cache sizes, even the entire set of files accessed by all users during the milestone might not fill the cache. Hence the response times would reach a plateau as the two cases e.g. 5 TB and 10 TB would be identical in terms of data storage. However, since the cache hits are around 50% in both the 5 and 10 TB cases, this explanation is not completely satisfactory.

To understand the results, a more detailed simulation was carried out. The simulation considered four cases, two of which are relevant to this case. These cases break down the MRT into its constituent parts i.e. time taken in the upper loop and lower loop of the PLM network. The upper loop is defined as the time taken by the PLM query to go from the Client to the Database and return to the Client. The lower loop refers to the time taken by the query to go from the Client to the FSC or Volume and return to the Client.

The tables A.1 and A.2 contain the results of these simulations. These tables show that despite the increase in CHR between the 5 TB and 10 TB cases, both the upper and lower loop MRTs remain the same. This implies that the increased 10% cache hits do not result in MRT reduction. The cause for this occurring is non-trivial and further establishes the fact that this system is too complex for using logical reasoning for drawing inferences, and that DES is the right approach. Possibly, the increasing cache hits do not decrease MRT as these queries would not have taken much additional more time in the lower loop, however, this is only conjecture.

5.2 Testing the Effect of Increasing Non-Milestone Users

Another useful scenario is to test the effect of increasing the number of non-Milestone users on the MRT, as it can show an organization the impact on the productivity of Milestone users due to the loading of the PLM network with other users during a milestone. The response times of the different user roles were separately calculated, as the different users work with files of different sizes, and the users working with larger files may

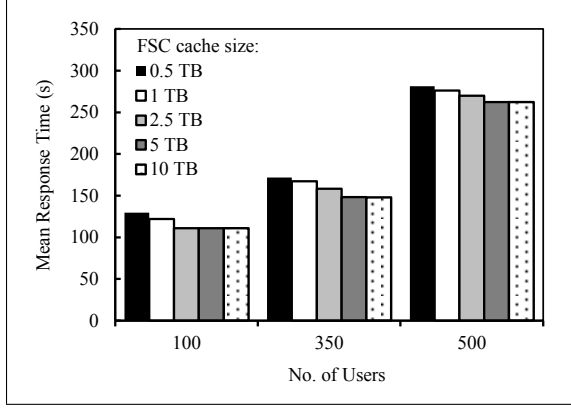


Figure 5.1: Response time (in s) for various cache sizes and number of users

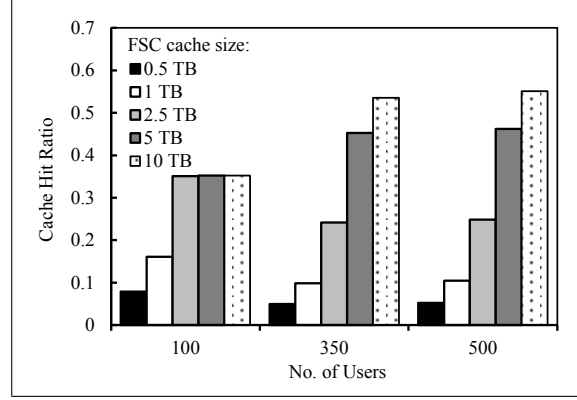


Figure 5.2: Cache Hit Ratio for various cache sizes and number of users

be more affected with the increase in non-milestone user in the system, as compared to those working with smaller files.

Results obtained show that increasing the number of software users increases the MRT. As previously stated, the additional time taken to respond to queries is not uniform between different user roles and scales with the average MRT for the different user roles. To explain this further, if only milestone users are working, we get a set of MRT numbers for different user roles i.e. in the present fig. 5.3, in increasing order of MRT, Engineer, Primary Designer, Manufacturing, Designer and CAE Analyst. As the number of users increase, the increase in MRT for the different user roles does not scale with an additive constant, but with a multiplicative one, i.e. the MRT for CAE Analysts is more drastic than that of the Designers is more drastic than that of the Manufacturing and so on.

Results from fig. 5.3 and from table A.3 in the appendix show that if two hundred non-Milestone users start working alongside the one hundred Milestone users, it can result in a 15% – 30% increase in response time. As the number of non-milestone users increases to three hundred, it results in a 50% – 75% increase in response time. Finally, if the number of non-milestone users increases to four hundred, it can result in a 100% – 200% increase in response time. While the 100% increase is seen for the CAE Analyst, and the 200% increase is seen for the Primary Designer and Engineer, the overall magnitude of increase is much worse for the CAE Analysts, as they have to access very large files over the network. As shown in the figure, the CAE Analyst may have to wait for as much as 15 mins. before that operation he/she was carrying out completes.

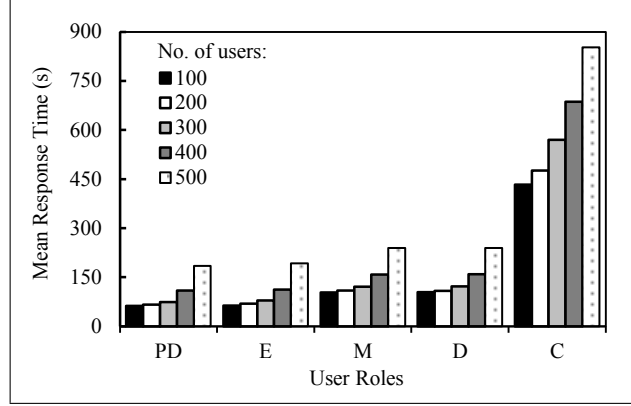


Figure 5.3: Response time (in s) for Primary Designer (PD), Engineer (E), Manufacturing (M), Designer (D) and CAE Analyst (C) and varying number of Non-Milestone users

5.3 Testing the Effect of Increasing the Starting Delay of Non-Milestone Users

In numerous cases, it is not possible to improve hardware parameters in a real industrial system, such as increasing the FSC cache size, as these actions would have monetary repercussions. A possible solution to this problem could be to request the non-Milestone users to start their operations fifteen minutes to half an hour after Milestone users start work. This would mean lower congestion in the system for Milestone users during start-up and a chance to populate the FSC cache with files relevant to Milestone users.

Results from fig. 5.4 show a significant difference between the response times of approximately 30% when the Non-Milestone users have a start delay time distributed between 0-15 mins. versus when they have the start delay time varying between 15-30 mins. Once again these results consider the various user roles separately. We observe that the decrease in MRT is most significant for CAE Analysts, since when the non-milestone users start 15 mins. late, instead of the MRT being approximately 15 mins. it decreases to 10 mins., a significant drop.

Fig. 5.5 and the tables A.4 and A.5 try to study this trend further to determine the cause for this improvement by once again looking at the upper and lower loops separately, and considers the MRT for users for the first 15 mins. separately from the rest of the time. The figure shows that the reduced Mean Response Time for Milestone users is observed both during the first 15 mins. and during the rest of the time. Due to lower congestion during the first fifteen minutes, as there is a vast difference in the Mean Response Time for the case when all users start together, as compared to the case when the Milestone users start early. However, as the Mean Response Time of Milestone users is also lower during the rest of the day, it implies that the preloading of the FSC cache in the first fifteen minutes also reduces the Mean Response

Time. This leads to another *What-if* scenario, where the FSC cache could be preloaded with Milestone user files.

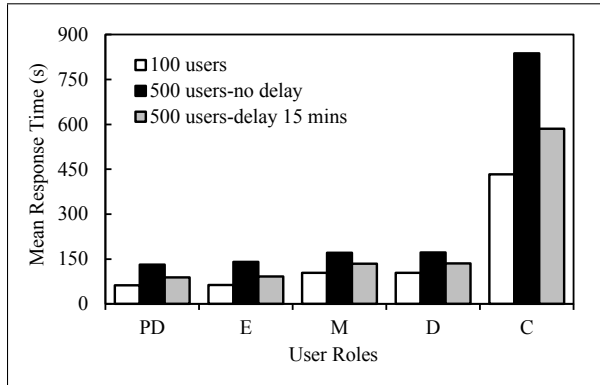


Figure 5.4: Mean response time (in s) for various roles and Non-Milestone users having a longer start time delay

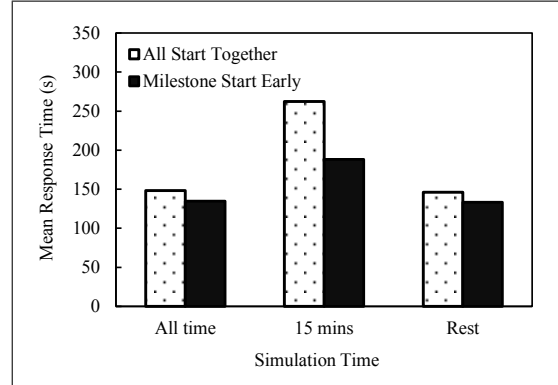


Figure 5.5: Impact of FSC cache size on Mean Response Time (in s) for Milestone users if the Non-Milestone users are initially delayed

5.4 Testing the Effect of Preloaded FSC Cache

As seen in section 5.3, preloading a cache seems to hold some promise in improving the MRT for milestone users. Thus, if a firm could preload the FSC cache with files that are essential to Milestone user operations, it could significantly increase Milestone user productivity.

Preloading the cache implies filling the cache with relevant files even before the simulation of the various users of the PLM software begin. With this in mind, two scenarios were tested. The first scenario preloaded the FSC cache with file that would assist the Milestone users launch their software faster. The second scenario preloaded the FSC cache with files for launching the software as well as data files that the Milestone users would require throughout the day.

Interesting results were observed, when these scenarios were compared against each other and the case with no preloaded cache. As seen in fig. 5.6 the Mean Response Time actually increased for both Milestone and Non-Milestone users when the cache was preloaded with only launch files. However, it halved for Milestone users, and decreased significantly for Non-Milestone users when the cache was preloaded with both launch and data files.

While the trends for MRT were surprising, the results for CHR were exactly as expected, as seen in fig. 5.7. For the case with preloaded launch data, as the data pertained only to milestone users, the CHR for milestone users is higher than the case for no preloading. On the other hand, the CHR for non-milestone users remains the same for both the case. For the third case, where both the launch and data files are

preloaded into the cache, we observe that the CHR is significantly higher for milestone users, which was expected. A surprising trend is that in this case, there is a synergistic effect that increases the CHR for non-milestone users too.

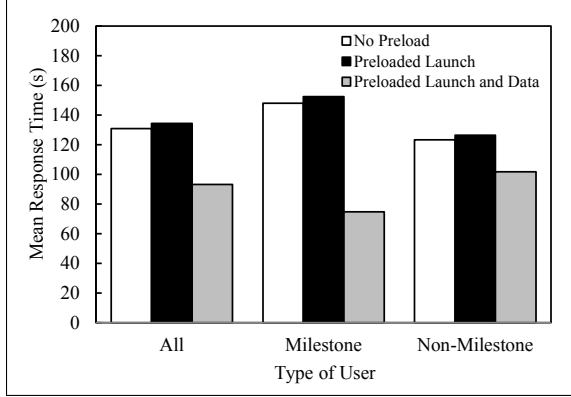


Figure 5.6: Mean Response time (in s) for users with Preloaded Cache

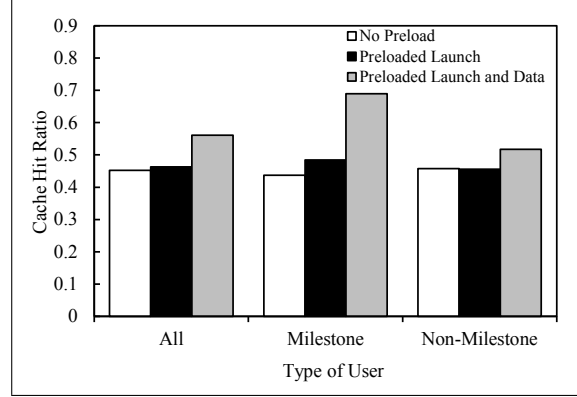


Figure 5.7: Cache Hit Ratio for users with Preloaded Cache

5.5 Testing Bottleneck at the Database

There are two points of bottleneck in the PLM network, the Database and the Volume. To observe the system response to bottlenecks, the number of queries that the Database can simultaneously handle was varied from one to ten and the resultant MRT values were observed. The FSC cache size was also varied to ensure that the observed trend was uniform. The results for the same have been plotted in figs. 5.8 and 5.9.

It was observed that as the number of messages handled by the Database increases, the MRT decreases. Fig. 5.8 plots the normalized trend to show that in all cases, as the cache size increases, MRT decreases, till a threshold value of around 5 TB, beyond which the MRT becomes stable.

This plot contains no information of the magnitude of MRT between different number of queries handled by the Database. That information is shown in fig. 5.9. This figure shows how the mean response time of the queries is affected by the number of queries. We can see that MRT is very high and unrealistic for the first 4-5 query numbers.

The results for very few messages handled by the database, namely one to six messages, have been omitted from fig. 5.8. This is due to the fact that the results did not provide the required statistical confidence. This has been further discussed in the simulation validation in section 6.

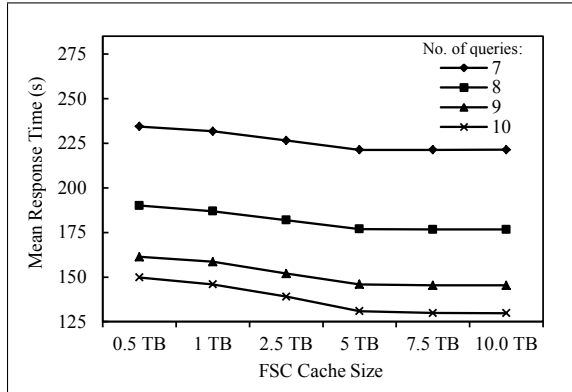


Figure 5.8: Mean Response Time Trend for varying number of queries handled simultaneously at the database

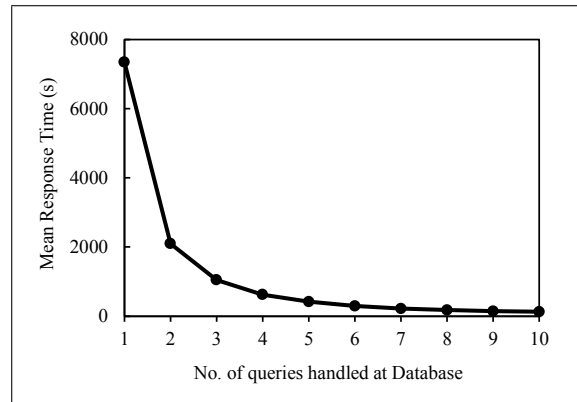


Figure 5.9: Variation of Mean Response Time (in s) with number of queries handled simultaneously at the database

Chapter 6

Validation

A crucial step in a simulation study is to validate the results being observed. In the present research, two forms of validation have been performed. The first form, model validation ensures that the results being observed represent the physical model being simulated. The second form of validation, statistical validation ensures that the results observed are statistically significant.

6.1 Model Validation

A number of different checks for model verification were carried out. These included manual observation of individual components of the PLM network to ensure each element was working separately, tracing the paths of messages to ensure that the different components combined functioned as expected and monitoring values from the running code to ensure that the values were feasible. One such test explained below shows that the different types of actions performed by users in the code corresponds to the expected values.

Table 4.4 in section 4.4 explained how each user working with the PLM software is performing one of three activities, Design checking, Issue Resolution and Design Release. Each of these activities involve a series of actions such as launch, search, read and write. Considering the case of five hundred simultaneous users of the PLM software, of which one hundred are Milestone users, the number of users working on each kind of activity was calculated. Using these values, we calculated the approximate ratio of launch, search, read, write, workflow and notify queries, and found that there should be four workflow queries for each other form of query. This was plotted at various times of the simulated day. As seen in fig. 6.1 the observed output from the code matches very well with the expected ratio of actions that we had calculated.

6.2 Statistical Validation

Since each run of the PLM simulation uses random number generators numerous times, the results obtained from the code are random variables over unknown distributions. Hence statistical tools have been used to ensure that the trends and results observed are repeatable and valid.

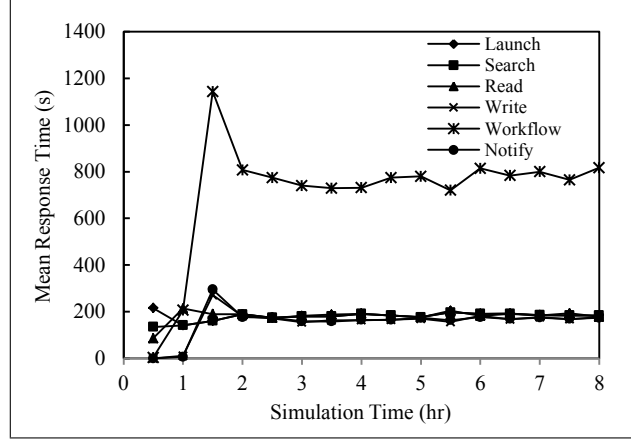


Figure 6.1: Number of queries for different operations at various times of the simulation

6.2.1 Number of Runs of the Code

To ensure the validity of the results, for every set of parameters, the code is run numerous times, with different seed values of the random number generator, and the results are averaged over these multiple runs. The Mean Response Time for all users is plotted in fig. 6.2, averaged over i runs where i is the number of runs that have been completed. Thus when this graph shows a stable value, the x label value indicates the number of runs that is required for a satisfactory results. As can be seen in the figure, the results are sufficiently stable after a hundred runs. Hence this is the value that has been chosen for the number of runs.

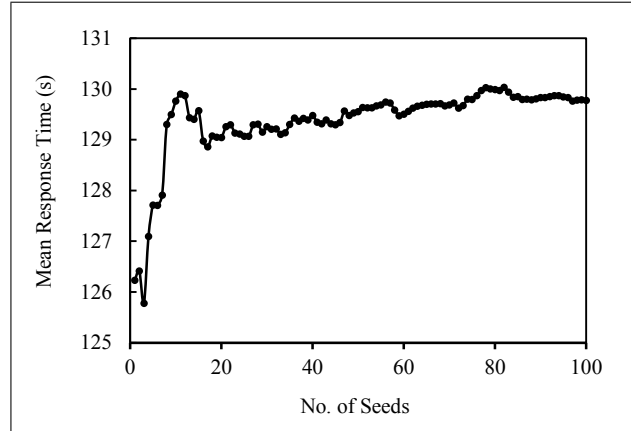


Figure 6.2: Mean Response Time (in s) averaged over completed simulations versus the number of completed simulations

6.2.2 Confidence Intervals

In section 5.5, results for low values of ‘queries handled by the database’ were not considered. The reason for this was that the trends that the results showed were not statistically significant. For example, in the case where ‘queries handled by the database’ is one, the Mean Response Time for a query was around 7000 s. Considering that the simulation was conducted for an eight hour work day, this implied that the Mean Response Time was calculated over three or four numbers. Comparatively, for the case where ‘queries handled by the database’ is ten, the Mean Response Time is around 130 s, which implies that it is an average over approximately 220 numbers. Clearly the second result would be more accurate than the first.

To validate this logical argument, 95% confidence intervals were evaluated for each of the two cases using the approach given in section 6.2.2. The results are shown in figs. 6.3 and 6.4. For the one query case, the confidence intervals are so large that drawing conclusions from the data is not possible. However, in the case with ten queries, the confidence intervals are narrow. Hence the trends observed have statistical significance.

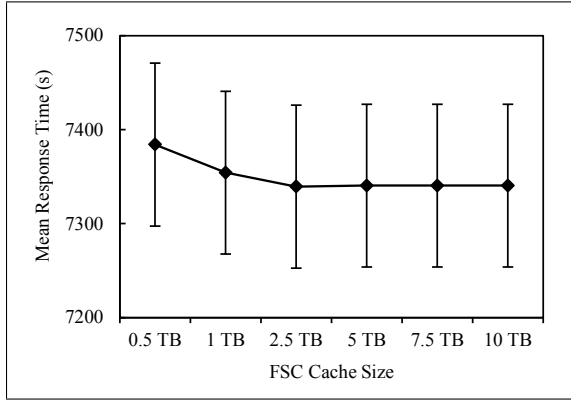


Figure 6.3: Mean Response Time (in s) and its 95% confidence interval for the case with 1 query handled by the Database

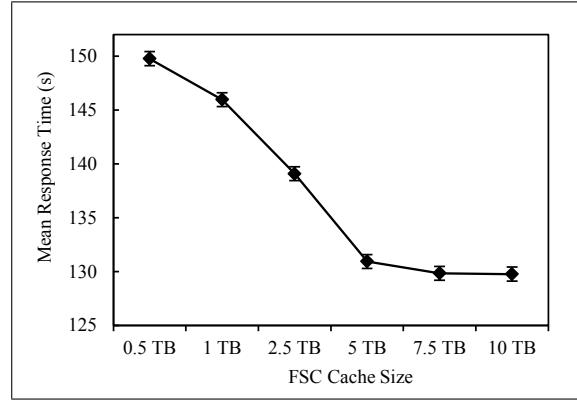


Figure 6.4: Mean Response Time (in s) and its 95% confidence interval for the case with 10 queries handled by the Database

Confidence Interval Calculations

In the previous section, section 6.2.1, it was decided that each simulation would be carried out 100 times. Thus MRT is calculated from eq. 6.2.1.

$$\bar{R} = \sum_{i=1}^n \left[\frac{R_i}{n} \right] \quad (6.2.1)$$

Where R_i is the i^{th} MRT and n is 100.

The sample variance S^2 was calculated for all the runs with different parameters, using eq. 6.2.2.

$$S^2(n) = \frac{1}{n-1} \sum_{i=1}^n (R^i - \bar{R}(n))^2 \quad (6.2.2)$$

Where n is 100, $\bar{R}(n)$ is the MRT averaged over n samples and R^i is the i^{th} MRT.

The 95% confidence interval was then calculated for the entire data set, using eq. 6.2.3.

$$\text{Interval} \in \left[\bar{R}(n) - t_{n-1, \frac{\alpha}{2}} \times \frac{S(n)}{\sqrt{n}}, \bar{R}(n) + t_{n-1, \frac{\alpha}{2}} \times \frac{S(n)}{\sqrt{n}} \right] \quad (6.2.3)$$

Where n is 100, α is 0.05, and \bar{R} and $S(n)$ denote the average of MRTs across samples and the sample standard deviation respectively.

The ‘t-value’ was calculated as ± 1.984 for a 95% confidence interval, and 99 degrees of freedom.

Chapter 7

Conclusions

The present study models and simulates the data storage network of a PLM software with the aim of improving the responsiveness of the software to users who are in critical phases of projects while using it. The developed model characterizes the PLM network as containing three components, namely hardware, software and users actions. Queuing and caching models are used to represent the hardware component, while for the software component, software commands have been represented as a series of queries, with each query having a number of parameters to represent the key differences between various commands. The user actions presented in this study represent the actions of a product design team during milestone operation for a manufacturing firm. The actions are modeled as software query sequences, with user parameters to model the myriad operations that different users of a software might perform. The study shows that hardware modification such as increasing the FSC cache size only decreases the mean response time by around 5 – 8% even though the cache hit ratio increases almost five fold. However, there are a number of operational modifications that show much greater improvement in performance. Delaying the starting time of non-milestone users by as little as fifteen minutes can improve the mean response times of milestone users by as much as 30%, while preloading the FSC cache with data essential to milestone users can improve the mean response time for milestone users by almost 50%. Thus, as per our study, companies using networks infrastructure which is similar to the infrastructure modeled in the present study might actually be able to improve the network response much more by innovative operational modifications to their user actions rather than simply network hardware upgrades.

Appendix A

Result Tables

Table A.1: Mean Response Time breakdown for FSC cache = 5 TB and all users starting together

FSC Cache Size		Mean Response Time (in s)		
5 TB				
All Users		Total	First 15 mins	Remaining Time
	All	130.94	217.45	130.00
	Top	32.33	70.49	31.58
	Bottom	98.64	170.68	97.86
	CHR	0.453	0.015	0.460
Milestone Users		Total	First 15 mins	Remaining Time
	All	148.07	262.25	146.05
	Top	36.77	76.64	35.39
	Bottom	111.33	209.44	109.59
	CHR	0.438	0.019	0.454
Non-Milestone Users		Total	First 15 mins	Remaining Time
	All	123.45	174.13	123.06
	Top	30.39	63.91	29.94
	Bottom	93.09	133.15	92.79
	CHR	0.457	0.011	0.462

Table A.2: Mean Response Time breakdown for FSC cache = 10 TB and all users starting together

FSC Cache Size		Mean Response Time (in s)		
10 TB				
All Users		Total	First 15 mins	Remaining Time
	All	129.77	217.45	128.83
	Top	32.52	70.49	31.77
	Bottom	97.26	170.68	96.47
	CHR	0.535	0.015	0.544
Milestone Users		Total	First 15 mins	Remaining Time
	All	147.74	262.25	145.71
	Top	37.01	76.64	35.64
	Bottom	110.73	209.44	108.99
	CHR	0.518	0.019	0.537
Non-Milestone Users		Total	First 15 mins	Remaining Time
	All	121.95	174.13	121.55
	Top	30.56	63.91	30.12
	Bottom	91.39	133.15	91.08
	CHR	0.540	0.011	0.546

Table A.3: Percentage increase in MRT for Primary Designer (PD), Engineer (E), Manufacturing (M), Designer (D) and CAE Analyst (C) and varying number of Non-Milestone users

		% Increase in MRT for:				
No. of Clients (milestone + non-milestone)		PD	E	M	D	C
100+100		6.33	8.07	5.04	4.34	9.90
100+200		19.34	24.49	16.84	16.74	31.62
100+300		74.72	77.20	52.87	52.94	58.50
100+400		195.89	201.78	130.44	129.65	96.68

Table A.4: Mean Response Time breakdown for FSC cache = 5 TB and non-milestone users starting 15 mins late

FSC Cache Size		Mean Response Time (in s)		
5 TB				
All Users		Total	First 15 mins	Remaining Time
	All	123.82	188.17	123.24
	Top	32.69	49.02	32.45
	Bottom	91.15	154.51	90.59
	CHR	0.451	0.031	0.456
Milestone Users		Total	First 15 mins	Remaining Time
	All	134.72	188.17	133.13
	Top	35.68	49.02	35.02
	Bottom	99.06	154.51	97.41
	CHR	0.435	0.031	0.454
Non-Milestone Users		Total	First 15 mins	Remaining Time
	All	118.99	-	118.99
	Top	31.36	-	31.36
	Bottom	87.65	-	87.65
	CHR	0.457	-	0.457

Table A.5: Mean Response Time breakdown for FSC cache = 10 TB and non-milestone users starting 15 mins late

FSC Cache Size		Mean Response Time (in s)		
10 TB				
All Users		Total	First 15 mins	Remaining Time
	All	122.26	188.17	121.67
	Top	32.85	49.02	32.61
	Bottom	89.41	154.51	88.83
	CHR	0.533	0.031	0.539
Milestone Users		Total	First 15 mins	Remaining Time
	All	133.65	188.17	132.03
	Top	35.86	49.02	35.21
	Bottom	97.79	154.51	96.10
	CHR	0.516	0.031	0.539
Non-Milestone Users		Total	First 15 mins	Remaining Time
	All	117.24	-	117.24
	Top	31.52	-	31.52
	Bottom	85.72	-	85.72
	CHR	0.538	-	0.538

References

- [1] M. McLuhan, *Understanding Media: The Extensions of Man*, ser. Routledge classics. Routledge, 2001.
- [2] CIMdata, “Product lifecycle management-empowering the future of business,” CIMdata, Inc., Tech. Rep., October 2002, [Online]. Available: http://www.cimdata.com/en/component/docman/doc_download/2523-product-lifecycle-management-plm-definition [Accessed: Feb. 27, 2014].
- [3] D. Dutta and J. P. Wolowicz, “An introduction to product lifecycle management (plm),” in *In Proceedings of the 12th ISPE International Conference on Concurrent Engineering: Research and Applications, Fort Worth/Dallas*, 2005, pp. 25–29.
- [4] E. Miller, “Manufacturers go digital to stay competitive,” CIMdata, Inc., Tech. Rep., November 2006, simPy. [Online]. Available: <http://www.cimdata.com/images/Articles/Harvest> [Accessed: Feb. 27, 2014].
- [5] (2014, April) Siemens plm teamcenter. Siemens PLM. [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/teamcenter/index.shtml [Accessed: Feb. 27, 2014].
- [6] Agile PLM, Oracle. [Online]. Available: <http://www.oracle.com/us/products/applications/agile/product-lifecycle-management/overview/index.html> [Accessed: Feb. 27, 2014].
- [7] L. Libman and A. Orda, “The designer’s perspective to atomic noncooperative networks,” 1999.
- [8] A. N. Tantawi, G. Towsley, and J. Wolf, “Optimal allocation of multiple class resources in computer systems,” *SIGMETRICS Perform. Eval. Rev.*, vol. 16, no. 1, pp. 253–260, May 1988.
- [9] T. Abdelzaher, K. G. Shin, and N. Bhatti, “Performance guarantees for web server end-systems: A control-theoretical approach,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, p. 2002, 2001.
- [10] M. Ahmad, A. Abounaga, S. Babu, and K. Munagala, “Modeling and exploiting query interactions in database systems,” in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ser. CIKM ’08. New York, NY, USA: ACM, 2008, pp. 183–192.
- [11] D. Villela, P. Pradhan, and D. Rubenstein, “Provisioning servers in the application tier for e-commerce systems,” in *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on*, June 2004, pp. 57–66.
- [12] L. P. Slothouber and P. D., “A model of web server performance,” in *In Proceedings of the Fifth International World Wide Web Conference*, 1996.
- [13] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, “An analytical model for multi-tier internet services and its applications,” in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS ’05. New York, NY, USA: ACM, 2005, pp. 291–302.

- [14] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 3, no. 1, pp. 1:1–1:39, Mar. 2008.
- [15] D. McWherter, B. Schroeder, A. Ailamaki, and M. Harchol-Balter, "Priority mechanisms for oltp and transactional web applications," in *Data Engineering, 2004. Proceedings. 20th International Conference on*, March 2004, pp. 535–546.
- [16] J. Ellithorpe, Z. Tan, and R. Katz, "Internet-in-a-box: Emulating datacenter network architectures using fpgas," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, July 2009, pp. 880–883.
- [17] G. Soundararajan and C. Amza, "Autonomic provisioning of backend databases in dynamic content web servers," in *In: Proceedings of the 3rd IEEE International Conference on Autonomic Computing (ICAC, 2005*, pp. 231–242.
- [18] N. Tallent, J. Mellor-Crummey, L. Adhianto, M. Fagan, and M. Krentel, "Diagnosing performance bottlenecks in emerging petascale applications," in *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, Nov 2009, pp. 1–11.
- [19] G. Ren, E. Tune, T. Moseley, Y. Shi, S. Rus, and R. Hundt, "Google-wide profiling: A continuous profiling infrastructure for data centers," *IEEE Micro*, pp. 65–79, 2010.
- [20] S. Herrero-Lopez, J. Williams, and A. Sanchez, "Large-scale simulator for global data infrastructure optimization," in *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, Sept 2011, pp. 54–64.
- [21] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. Das, "Mdcsim: A multi-tier data center simulation, platform," in *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, Aug 2009, pp. 1–9.
- [22] R. Buyya, R. Ranjan, and R. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsims toolkit: Challenges and opportunities," in *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, June 2009, pp. 1–11.
- [23] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation (3rd Edition)*, 3rd ed. Prentice Hall, 2000.
- [24] G. Fishman, *Discrete-Event Simulation: Modeling, Programming, and Analysis*, ser. Springer Series in Operations Research and Financial Engineering. Springer, 2001.
- [25] SimPy. [Online]. Available: <http://simpy.readthedocs.org/en/latest/> [Accessed: Feb. 27, 2014].
- [26] N. Matloff. (2008, Feb) Introduction to discrete-event simulation and the simpy language. [Online]. Available: <http://heather.cs.ucdavis.edu/matloff/156/PLN/DESimIntro.pdf> [Accessed: Apr. 10, 2014].
- [27] "C3P Data Complexity Reduction PSL Analytical Simulation," BIMCON Inc., Internal Tech. Report, February 2012.
- [28] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 338–354, 09 1953.