


August 2013

Economic Perspective on Cloud Computing: Three Essays

Abhijit Dutt

University of Wisconsin-Milwaukee

Follow this and additional works at: <https://dc.uwm.edu/etd>

 Part of the [Databases and Information Systems Commons](#), and the [Human Resources Management Commons](#)

Recommended Citation

Dutt, Abhijit, "Economic Perspective on Cloud Computing: Three Essays" (2013). *Theses and Dissertations*. 245.
<https://dc.uwm.edu/etd/245>

This Dissertation is brought to you for free and open access by UWM Digital Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UWM Digital Commons. For more information, please contact open-access@uwm.edu.

ECONOMIC PERSPECTIVE ON CLOUD COMPUTING: THREE ESSAYS

by

Abhijit Dutt

A Dissertation Submitted in

Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Management Science

at

The University of Wisconsin-Milwaukee

August, 2013

ABSTRACT

ECONOMIC PERSPECTIVE ON CLOUD COMPUTING: THREE ESSAYS

by

Abhijit Dutt

The University of Wisconsin-Milwaukee, 2013
Under the Supervision of Professor Hemant Jain

Improvements in Information Technology (IT) infrastructure and standardization of interoperability standards among heterogeneous Information System (IS) applications have brought a paradigm shift in the way an IS application could be used and delivered. Not only an IS application can be built using standardized component but also parts of it can be hosted by different organizations in different locations provided it can be accessed using the Internet. This dissertation is an attempt to uncover unique aspects of this phenomenon known as Software as a Service (SaaS).

The first essay examines design decision making by SaaS providers by analyzing effects of two non-functional attributes of an IS Application – modularity and architectural performance. We model the relationship of the two attributes with factors such as demand, price, and user’s preference. The model includes marginal cost and maintenance cost to recognize the service aspect of SaaS. Our

results show the optimal values of various decision variables while taking into account user's sensitivity to modularity, architectural performance and price.

The service component in cloud computing necessitates that the service providers plan for requisite delivery capacity. The second essay addresses optimal infrastructure capacity planning while taking into account the opportunity cost of having low capacity and cost of unused capacity in the case of high capacity. We develop a model which provides insight to a SaaS provider on optimal capacity planning of IT infrastructure when faced with a variable demand and performance expectations.

The third essay focuses on financial risks faced by SaaS providers in the context of provider's risk tolerance. We analyze the financial risk of provider's decision making on pricing, capacity and other factors that may lead to financial risk as they are based on incomplete information. We built a model using Mean Variance Analysis theory for investigating the effect of provider's risk tolerance on infrastructure capacity planning while taking into account modularity in software architecture and operational performance.

This dissertation extends our understanding of significant issues facing a SaaS provider. The models presented here can form the basis for an extensive

exploration of the phenomenon of SaaS specifically and Cloud Computing in general.

© Copyright by Abhijit Dutt, 2013
All Rights Reserved

This dissertation is dedicated to the memory of my mother Mrs.
Anima Dutt and my father Mr. Alok Kumar Dutt

ACKNOWLEDGEMENTS

This is probably the most enjoyable part of writing the dissertation. I would like to acknowledge the support of a number of individuals who made completion of this dissertation a reality. First, thanks go to my dissertation chair, Dr. Hemant K. Jain. Dr. Jain, you provided me with unconditional support while I struggled through my research. You have been a mentor to me, guiding me intellectually and personally during this wonderful as well as sometimes very frustrating journey. You gave me complete independence and at the same time you were there to help me whenever I needed. As I lived far away from Milwaukee, you made extra effort to guide me. You were always there whenever I needed any help.

I also would like to thank the other members of my dissertation committee- Dr. Huimin Zhao, Dr. Amit Bhatnagar, Dr. Sanjeev Kumar, and Dr. Sanjoy Ghose for providing valuable feedback on my dissertation. I also appreciate all your encouragement and support. Special thanks go to Dr. Kumar, you gave detailed, thoughtful and constructive comments that were extremely valuable. I really appreciate the amount of time you took to help me complete my dissertation.

There are a few individuals who helped in many different ways. First, I would like to thank Dr. Samar Mukhopadhyay. He introduced me to the area of theoretical modeling. Second, I would like to thank my fellow UW-Milwaukee doctoral alumni Dr. Robert Setaputra. Bob, I really appreciate many discussions we had over coffee and your support and help in this research.

I would like to acknowledge the two most important girls in my life. My daughter Bela was born while I was working on my dissertation. Her smile and love enabled to me to forget all the pressure and frustration; I know you are looking forward to spending more time with me. Finally, I want to acknowledge my wife Paula; without her love and unwavering belief in me I would not have been able to finish this dissertation. I really appreciate your sacrifice and I know you are looking forward to the day when I will not be working on my dissertation any more. You patiently endured many long hours alone, took care of our daughter Bela by yourself, moved to new areas twice while I worked on my dissertation. There are no words that can express my gratitude and appreciation for all you have done for me.

Table of Contents

Chapter 1: Introduction.....	1
Chapter 2: Essay One - Modularity and Performance in Cloud Computing: An Economic Perspective	11
2.1 Introduction	11
2.2 Motivation and Research Questions	12
2.3 Background and Previous Research	20
2.3.1 Cloud Computing	20
2.3.2 Modularity.....	24
2.3.3 Performance.....	27
2.4 Previous Research	29
2.5 Model Formulation:.....	35
2.6 Results:.....	45
2.6.1 Case 1: Modularity (m) and Architectural Performance (s) are not related	45
2.6.2 Case 2: Modularity (m) and Performance (s) are related.....	55
2.7 Discussions of Results and Implications.....	66
2.8 Limitations.....	68
2.9 Future Directions.....	68
Chapter 3: Essay Two - Demand Planning for Cloud Computing: Effect of Random Variation in Demand.....	70
3.1 Introduction	70
3.2 Motivation and Research Questions	71
3.3 Literature Review	75
3.4 Model Formulation.....	78
3.5 Theoretical Results	82
3.6 Numerical Results:.....	93
3.7 Discussions	96
3.8 Limitations:.....	96
3.9 Conclusions and Contributions:	97
3.10 Future Directions.....	98
Chapter 4: Essay Three - A Financial Risk Model for Cloud Computing	99
4.1 Introduction:	99
4.2 Introduction to Risk Management	102
4.3 Literature Review	105
4.4 Capacity Planning Framework:.....	106

4.5	Modularity and Model Formulation	107
4.5.1	Modularity.....	107
4.5.2	Model Formulation	110
4.6	Results.....	124
4.7	Limitations.....	128
4.8	Future research	129
Chapter 5: Contributions.....		131
5.1	Research Contributions	131
5.2	Contributions to Practice	133
Bibliography.....		135

List of Figures

Chapter 2:

Figure 2.1: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to customer sensitivity to modularity in the IS application architecture ... 50

Figure 2.2: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to change in fixed cost for introducing modularity into the system 53

Figure 2.3: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to change in sensitivity to modularity..... 61

Figure 2.4: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to change in maximum architectural performance 65

Chapter 3:

Figure 3.1: Graph showing the relationship between optimal capacity (Q) and ω 94

Figure 3.2: Graph showing the relationship between Average profit and Standard deviation of profit with respect to capacity 95

Chapter 4:

Figure 4.1: Graph showing the effect of change in planned capacity on Average profit for risk neutral (red line) and risk averse (blue line) providers and Standard deviation of profit 125

Figure 4.2: Graph showing the relationship between optimal capacity and risk averseness factor ε for different values of ω 127

Chapter 1: Introduction

Wide acceptance of computing and computing enabled artifacts in everyday life is changing the way we live. As an example, we are seeing an explosive growth in the adoption of Smartphones, tablet computers etc. These devices are fundamentally different from traditional computers. First, these devices are able to access the Internet using mobile networks. Second, these devices run on many different types of operating systems. Third, these devices have less memory and slower processors. However, the traditional computers are still available and are used and in most cases they run on different operating system platform.

Wide use of different end user devices that ranges from traditional computers to smartphones makes it imperative for providers of cloud applications to develop IS applications in such a way that a user could access same IS applications using many different types of hardware from different locations that are outside the enterprise (Yoo, Henfridsson, & Lyytinen, 2010). We are also observing wide adoption and use of social networking sites such as LinkedIn, Facebook, and MySpace for personal as well as for business use. The advent of services such as Skype, Google Hangout is also changing the way we make telephone calls and it is opening up different innovative ways to communicate; it is no longer necessary to have expensive specialized

equipment for video conferencing. People are writing and sharing documents using different services such as Google Doc, dropbox using the Internet. From a technological point of view we can identify the following two reasons that are responsible for the above mentioned innovations. First, there have been significant improvements in Information Technology (IT) and networking infrastructure such as availability of very high speed Internet connection. Second, standardization of protocols such as SOAP has made it possible for different IS applications to interoperate with one another seamlessly. Hence, it is no longer necessary to have computing restricted within an enterprise. It does not matter where software and hardware is located, a user or an Information System (IS) application can access the necessary resources (another IS application, other software, hardware etc.) as long as those resources are connected to the Internet (Carr, 2005). This phenomenon is known by many different names such as cloud computing, service oriented computing, utility computing (Vaquero, Rodero-Merion, Caceres, & Lindner, 2009). It is a great opportunity for IS researchers to understand and study this complex evolving phenomenon; however such investigations require a new perspective (Yoo, 2010). This dissertation is an attempt to accomplish that.

Success of IS applications in organizations have been studied using two main perspectives – the users' perspective and the developers' perspective. Taking users' perspective, acceptance of IS application has been studied focusing on

different themes; however, there is a commonality among these themes; it was hypothesized that success or failure of an IS application (or IS innovation) depended on the organization's attribute (Fichman, 2004). Taking the perspectives of developers, researchers in the areas of computer science and Information Systems studied IS applications focusing mainly on development methodologies, processes, modeling etc. The established areas of "Software Engineering" and "System Analysis and Design" are devoted to this area of research. Uncovering of functional attributes of a proposed IS application has been considered one of the most important steps during development of an IS application. These approaches were appropriate, as most IS applications were built to solve very specific business problems. Hence, it is fair to conclude that most IS applications were treated as customized products. However, cloud computing applications need to be more general in nature so that these applications are useful to diverse group of users. Hence, it is no longer possible to assume that the IS applications are customized products.

According to National Institute of Standards and Technology (NIST) guideline published in 2011, "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011). In

this definition, we note the use of word "service" and the word "configurable". The word "configurable" does have a very important implication. It is not necessary to build a configurable IS application; unless it is envisaged that many different customers will use it. Hence we conclude, unlike a traditional IS application which is a customized product, an IS application in a cloud computing domain is a combination of product (not customized) and service (Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010). It is very likely, in future most businesses will purchase Information Technology (IT) services in the same way they purchase telephone services and there will be little need for businesses to invest on real IT asset such as purchasing hardware and software. Not only it is a fundamental change for every user of IS applications but also it has a major impact on developers of IS applications. We noted earlier that most traditional IS applications were customized products. However, cloud enabled IS applications need to be useful to many different types of users. Hence, it is fair to conclude that cloud computing is changing a typical IS application from a customized product to mix of (non-customized) product and service. This phenomenon has been described as industrialization of IT. In this dissertation we study the effects of industrialization of IT from the perspectives of developers of IS applications.

The term cloud computing is actually an umbrella term; there are many different types of cloud computing solutions. Cloud computing could be

classified into three different types of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (Fouquet, Niedermayer, & Carle, 2009; Sridhar, 2011; Vaquero, Rodero-Merion, Caceres, & Lindner, 2009). A typical IaaS provides low-level computing resources using hardware virtualization technologies. It gives customers full control over the operating systems and installed applications. Amazon EC2 is an example of an IaaS (Amazon.com). A typical PaaS exposes appropriate Application Programming Interfaces (API) that can be used by application developers to create applications. Google App Engine is an example of a PaaS (Google). A typical SaaS provides users direct access to an IS application. Salesforce.com's Sales Cloud is an example of a SaaS (Salesforce.com). In this dissertation, we focus on development of SaaS applications. In the next chapter, we discuss in detail different types of cloud computing and also why this dissertation focuses on SaaS applications.

The number of companies who are offering SaaS solutions is growing rapidly and most established computing companies such as Microsoft, SAP have also entered this space. The SaaS companies are also increasing their offerings. A quick look at the customer lists of different SaaS providers such as Amazon, Google, Salesforce.com, and Microsoft shows a rapid increase in their customer base. As we discussed above, we recognize that a cloud computing IS application such as a SaaS application is a mixture of product and service. In

order to understand and analyze this new development it is necessary to take a look at some nontraditional IS areas such as service science (Demirkan, Kauffman, Vayghan, Fill, Karagiannis, & Maglio, 2008) and new product development (Nambisan & Wilemon, 2000).

In the marketing literature product is defined as an entity that has complex bundle of attributes which provide core benefits, tangible benefits and intangible benefits to the consumers (Krishnan & Ulrich, 2001). We use this definition of product, and we define an IS application as an entity that has many attributes. The attributes which are directly related to the functionality of the application are defined as functional attributes. The attributes that are not directly related to the functionality of an IS application are defined as non-functional attributes. In the IS discipline, non-functional attributes such as modularity, performance, security have not received sufficient attention (Chung & Sampaio do Prado Leite, 2009). As a typical IS application is morphing into a product service, it is no longer possible to ignore the important roles the non-functional attributes could play for determining the efficacy of a SaaS application. Service could be defined as a relationship between a producer and a consumer that creates and captures value and where the consumer participates actively (Gadrey, 2000; IBM). In other words, in the case of services, the consumers could be considered as co-producers (Fitzsimmons & Fitzsimmons, 2004).

Based on previous discussions we conclude that there are some fundamental differences between a traditional IS application and a cloud computing enabled IS application such as a SaaS application. Taking a perspective of SaaS application developers, we develop an analytical framework that would help managers make different decisions during development of SaaS applications. This dissertation is formatted as three essays and we summarize them here.

In the first essay, we investigate the role two important non-functional attributes of IS applications – modularity and performance in software architecture play on optimal profit, price and demand. Using unconstrained optimization, we model the profit function that includes marginal and maintenance costs. First, we consider that modularity and performance in software architecture are independent of each other. Second, we assume that there is a relationship between them; increase in modularity leads to decrease in performance in software architecture.

In the case of a traditional IS application in an enterprise setting, it is the customer's responsibility to arrange for the infrastructure such as hardware and networking devices and to install and maintain the IS applications. However, in the context of a SaaS application the relationship between a producer and consumer is different because of the service aspect of a SaaS application. It is the developer's (SaaS provider's) responsibility to arrange for infrastructure so

that SaaS could be consumed by its customers. Developers of SaaS applications need to make provision for necessary hardware and software components so that they can offer their services at an acceptable level to their customers. The second essay focuses on capacity planning. Here, we investigate how SaaS application providers can determine the necessary optimal capacity for their applications.

The demand of a SaaS application could be predicted on the basis of price and other attributes such as operational performance. However, it is likely that there would be random variation in demand that cannot be predicted. So a SaaS application provider could face two different scenarios. In the first case, the demand is higher than the anticipated demand. In that case, a provider would not be able to satisfy all the potential customers and would lose potential revenue. This loss might become more costly to a provider as some of its potential customers might decide to go with competitors for all their future needs. In the second case, the demand is lower than the anticipated demand. In that case, the provider will not be able to use all its capacity and will unnecessarily incur cost for infrastructure that will not be used.

In order to accurately plan for capacity, it is necessary to incorporate random variations in demand in the model. In this essay, we take a two-step innovative approach for demand prediction. First, we calculate optimal price and demand

using profit maximization model for vendors. Second, we model a stochastic profit maximization problem where demand follows a Gaussian probability distribution function with a mean optimal demand as calculated in the previous step. We assume that there is no change in price because of random variation in demand, and the producers will charge the optimal price that was obtained in previous step. We investigate how the optimal capacity varies under various circumstances. We also investigate the role operational performance plays in this context.

In the second essay, we introduced importance of capacity planning for the providers of SaaS applications and we presented a method to calculate optimal capacity. However, one size fits all methodology may not be appropriate for all. There will be providers who would prefer to plan for lesser capacity so that the probability of loss from excess capacity is less. On the other end of the spectrum there will be providers who would like to have larger capacity so that they will not lose out in case of higher demand. In addition to this uncertainty, prior research has observed that IT investments in general are riskier than other capital investments (Dewan, Shi, & Gurbaxani, 2007). The third essay investigates how financial risk tolerance of SaaS application providers would affect capacity planning for SaaS applications. Using Markowitz's mean variance analysis model (Markowitz, 1959), we build a financial risk model so that the SaaS application providers could make appropriate decisions based on their

individual risk tolerance. We shall also investigate the relationship of financial risk tolerance with expected (average) values of different decision parameters.

Chapter 2: Essay One - Modularity and Performance in Cloud Computing: An Economic Perspective

2.1 Introduction

Cloud computing is fundamentally changing the way computing resources are developed, provided and used. Some of the observed changes are how an IS application is accessed; how the applications are set up etc. Advent of networking in the early 70s made it possible to connect many computing devices together. As a result of that it was possible for businesses to network smaller powered computers and use them instead of using a bigger and more powerful mainframe computer. It was no longer necessary to replace a computer when there was a need for higher computing power; another computer could be added to the enterprise. It gave rise to client server computing, that we are still using today. It made computing scalable and more efficient. Cloud computing has already changed computing significantly and this trend will continue. In this essay, first we identify some of the important changes that are occurring because of the advent of cloud computing. Second, using unconstrained optimization, we find relationships among price, modularity and architectural performance of a SaaS application under the condition of profit maximization of its developer. The rest of the essay is organized in the following way. In the next section, we discuss in detail the motivation for this research. Next, we define cloud computing,

modularity and performance. A review of relevant literature is presented after this. We then formulate our profit maximization model which is followed by presentation of results of theoretical analysis. The last section discusses the results and presents possible future work in the area.

2.2 Motivation and Research Questions

In this section we discuss in detail how cloud computing is bringing a paradigm shift in computing. First, an IS application hosted in the cloud could be accessed from anywhere as long as there is a network connection. It is no longer necessary that users of an IS application are in the same enterprise where the IS application is hosted. Wide acceptance and use of different types of mobile computing devices such as smartphones, tablets is one of the effects of cloud computing.

Second, the concept of ownership of cloud computing applications is different from traditional computing applications. Instead of making outright purchase, businesses purchase Information Technology (IT) services in the same way they purchase different utility services such as telephone service, network connection service etc. As a result of the service component, delivery and use of cloud computing applications involves three distinct groups – developers of IS applications, service providers of IS applications and users of IS applications. Traditionally, the service providers and users were same as the

organizations installed the IS applications themselves and they owned the underlying IT infrastructure. In a cloud computing model, in many cases it is possible that same organizations could develop and provide the service; however the role of users of an IS application is significantly different. It is also important to note that in the cloud computing model the application provider and user will have a long term business relationship as the users are no longer purchasing the software outright.

Third, from a financial perspective as businesses start using cloud computing it is no longer necessary for them to make huge capital investment in IT infrastructure, such as hardware, software etc. The IS application providers either build their own IT infrastructure or they purchase it from another IT infrastructure service provider; in either case the users need not worry about the IT infrastructure issues. As an example, the social networking site Foursquare rents its IT infrastructure from Amazon.com.

Fourth, most traditional IS applications were built for solving a specific business problem of an organization. However, cloud computing enabled IS applications are no longer constrained to operate in a specific enterprise or a business; it becomes necessary that the developers of these IS applications build them focusing on a diverse group of users from a diverse group of businesses from many industries. As an example, the customer relationship

management (CRM) service provider Salesforce.com provides similar services to a diverse group of businesses. Hence, these IS applications need to solve problems in a generalized way; so it can be argued that unlike a traditional IS application which is a customized product, a cloud computing application is a combination of generalized (not customized) product and service (Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010). This phenomenon has also been described as industrialization of IT.

Traditionally software is considered a developmentally intensive product, because software has substantial development costs and small marginal cost for production (Krishnan & Zhu, 2006). However, cloud computing applications are different in that respect. Not only cloud computing applications have a substantial development costs but also a producer of cloud computing applications incurs a significant marginal cost of providing the service, because the consumer producer relationship could last throughout the lifetime of the product.

Although Cloud computing has received lot of attention in the industry, there are very few academic articles on cloud computing. Choudhary compared how software quality could be different under perpetual licensing scheme (traditional software product) and Software as a Service (SaaS) (Choudhary, 2007). However, he did not define software quality and he implicitly assumed that meaning of software quality in the two scenarios were same. Zhang and

Seidman investigated the difference between the above two scenarios under quality uncertainty and network externality effects (Zhang & Seidmann, 2010). They defined software quality as an attribute with many different dimensions such as features, speed, functionalities etc.; their definition of software quality used the fact that software quality could be defined in terms of functional attributes (Agrawal & Chari, 2007) as well as in terms of non-functional or structural attributes (Capra, Francalanci, & Merlo, 2008) of software. Although there is a rich literature on software quality, the meaning of software quality has changed over time (Agrawal & Chari, 2007). Recently, the Software Engineering Institute (SEI) at Carnegie Mellon University, and the Object Management Group have jointly formed a Consortium of Software Quality (CISQ) to investigate issues in software quality. The group's main focus is to devise metrics which could be used for measuring software quality; hence it is fair to conclude it is difficult to measure software quality accurately. Hence, in this dissertation we focus on tangible nonfunctional attributes.

In order to understand and analyze this new development it is necessary to take a look at some nontraditional IS areas such as service science (Demirkan, Kauffman, Vayghan, Fill, Karagiannis, & Maglio, 2008). Although literature has recognized the service aspect of cloud computing, to our

knowledge there is no analytical model for cloud computing applications which include any specific service characteristics such as marginal or maintenance cost of the service. Most researchers have treated cloud computing applications as developmentally intensive products (Krishnan & Zhu, 2006). In this essay, we intend to address the specific gaps in research as indicated above.

In this research, we focus on non-functional attributes of a cloud computing application. Although, we recognize that functional attributes are the most important characteristics of any product, the functional attributes are specific to a particular product. On the other hand, non-functional attributes such as quality and usability are applicable to all products. Also in a cloud computing model, it is possible that many providers will offer similar service in terms of functional attributes. However those providers could compete by offering different levels of non-functional attributes. Hence, we only include non-functional attributes in our model, and as a result of that our model is relevant to any cloud computing applications. It will enable us to uncover important insights into a typical cloud computing application and it will help us understand the phenomenon of cloud computing and uncover important common features among different types of cloud computing applications.

We focus our attention on specific and meaningful non-functional attributes of IS that could be operationalized in future; however operationalization of attributes is beyond the scope of this essay. Our work is different from the other works in the cloud computing area in the following ways. First, we focus on two important and specific non-functional attributes of an IS – modularity and performance which have been identified as important dimensions of software quality (Joglekar & Rosenthal, 2003; Agrawal & Chari, 2007). Second, we recognize the service aspect of cloud computing applications by including both marginal production and recurring cost of providing the service in the model.

Modularity is considered to be one of the most important non-functional attributes of an IS application (Parnas, 1972; Parnas, Clements, & Weiss, 1985). We define and discuss modularity in a following section. It has been observed that there is an optimal level of modularity in an IS application; too little or too much modularity could be detrimental, because increase in modularity in some cases may reduce the performance of an IS application (Banker, Datar, Kemerer, & Zweig, 1993). In a recent commentary, Yoo et al have opined that recent developments necessitate considering the importance of modularity in software architecture (Yoo, Henfridsson, & Lyytinen, 2010).

Performance is another important non-functional attribute. Performance is always considered an important part of Service Level Agreements (SLA). In the cloud computing area, SLAs are important as they formalize the relationship of service provider and service consumer (Demirkan, Kauffman, Vayghan, Fill, Karagiannis, & Maglio, 2008).

Cloud computing could be classified into three different types of services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (Fouquet, Niedermayer, & Carle, 2009; Sridhar, 2011). We provide a detailed definition of all of them in a following section. In this essay, we focus on SaaS applications because of the following reasons. First, a SaaS application is similar to traditional IS application and the consumers of such applications are very similar to traditional IS application users. Second, a SaaS application developer faces some of the similar challenges as a traditional IS application developer. Hence, focusing on SaaS will help us uncover and study the paradigm shift cloud computing is bringing to the area of IS application. We investigate specifically the roles of modularity and performance in a SaaS application architecture. Taking the perspective of a SaaS application provider we investigate the following three research questions.

1. While developing a SaaS application, what are the optimal levels of price, performance and modularity that will lead to maximum profit for software developers? How are the above three attributes related to one another?
2. How does a change in SaaS application users' sensitivity to different parameters such as price, modularity and performance affect optimal values of demand and profit under the condition of profit maximization?
3. What are the effects of different costs incurred by the cloud computing application providers such as fixed cost, marginal cost and maintenance cost on different parameters such as price, demand and modularity under the condition of profit maximization of the producers?

We do the above analysis under two scenarios. In the first scenario, we assume that there is no relationship between performance and modularity. In the second scenario, we assume an inverse relationship between performance and modularity as prior research has shown that increase in modularity leads to decrease in performance (Clark, 1982).

2.3 Background and Previous Research

As cloud computing is a new phenomenon, the definitions of different concepts related to this area are still evolving. In this section, we review the available definitions of cloud computing, SaaS, modularity and performance.

2.3.1 Cloud Computing

National Institute of Standards and Technology (NIST) published a guideline in 2011 that included a definition for cloud computing. The definition states that "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011). There are a few more definitions of cloud computing in the literature. According to (Vaquero, Rodero-Merion, Caceres, & Lindner, 2009) cloud computing could be defined as "Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs". In this essay, we use the NIST definition of cloud computing.

From the above definitions, we observe that cloud computing involves a service provider and consumers of services. The service provider provides different types of computing services, such as computing infrastructure, computing platform or just an IS application. The consumer uses the above services according to specific Service Level Agreements (SLA). The service could be offered at different levels of abstraction depending on the type of specific access and control a consumer has. So far three distinct abstractions have been identified and they are discussed below.

2.3.1.1 Infrastructure as a Service (IaaS)

These are services where consumers directly use infrastructure resources such as data storage, networking equipment, computer hardware from service providers. These services enable consumers to have lower level access to the IT infrastructure such as deploying software, changing hardware configuration. However the consumers are not responsible for the maintenance of the resources. Outsourcing of data center is an example of this. These types of services are known as IaaS. There are other alternative names such as utility computing. The first known reference to utility computing could be found in a lecture delivered by Dr. John McCarthy in 1961 at MIT. According to him "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

2.3.1.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) enables a user to develop, install and use an IS application using a cloud application provider's infrastructure. Although the developers have access to development environment in a providers' infrastructure; developers need not worry about any other infrastructure requirements. Some of the existing platforms which provide this type of services are Salesforce.com's Force, Microsoft's Azure, and Google's Apps Engine.

2.3.1.3 Software as a Service (SaaS)

In the literature, SaaS is defined as an IS application which is hosted in a vendor site and could be accessed by users through the Internet either using standardized protocols such as SOAP, REST or using proprietary protocols such as the one offered by Salesforce.com. SaaS enables a provider to offer IS application as a service to consumers by hosting the IS application in their own IT infrastructure; consumers need minimal IT infrastructure for consuming the service (Wikipedia). Salesforce.com's CRM application, Amazon's storage application and Google's Apps are examples of SaaS offerings. Here the consumers only need to focus on functionality of the application and they might be responsible for minimal configuration decisions.

As we can see all the three different types of service have many similarities.

First, in each case the consumers rent the needed functionalities instead of

either developing it in house or purchasing it outright. Second, consumers are not responsible for maintaining the underlying infrastructure necessary for using the service. The main difference among the three services is in the level of abstraction. First, SaaS is at the highest level of abstraction as the consumers are provided with access to an IS application. Second, the PaaS is at the middle level of abstraction; here the consumers are provided with access to application development environment. Third, IaaS is at the lowest level of abstraction; here consumers are provided with access to different lower level infrastructure necessary for computing. Taking another perspective, we observe that SaaS, PaaS, IaaS are similar to software applications, software development environments and basic computing infrastructure respectively; the main difference is that they are offered as services. This could lead to following possible scenario; it is possible for SaaS providers to develop SaaS applications using PaaS platforms from PaaS providers and then offer those applications as SaaS solutions to their customers. It is easy to see that use of cloud computing could make a supply chain of customers and vendors where vendor at one point is a customer at another. The outsourcing of a business's logistical functions such as transportation, warehousing, product returns to another organization is known as Third-party logistics (3PL or TPL); the organizations which provide this type of services are known as Third-party logistics providers or Third-

party service providers (3PSP) (Vitasek, 2010). The SaaS applications are somewhat similar to 3PL.

2.3.2 Modularity

Modularity is not a new concept. There is a long history of using modularity in product design. It is claimed that the Terracotta army figures in the mausoleum of the first Chinese emperor Qin Shi Huang were constructed using the principles of modularity in the third century BC. The head, arms, legs and torsos were created separately and then they were glued together.

Using the concepts of modularity, the Venetian Arsenal was able to produce about one ship a day during 16th century; they employed about 16,000 people (Wikipedia). The idea that an IS application should consist of many modules was first proposed by Parnas while introducing the concept of information hiding (Parnas, 1972). The concept of information hiding or encapsulation is based on the fundamental economic concept of division of labor, where tasks are divided into many tasks that could be performed by different people. However, especially in the area of intellectual division of labor, the effective coordination becomes an important issue and that takes away some of the benefit of division of labor. As an example, it has been observed that dividing a software product into modules required a significant effort in coordination and communication among the developers of different modules neutralizing some of the benefits of modularization. Hence it was

realized that in order to extract benefit from modularization, it is necessary to design modules in a way that required least amount of communication among the modules (Langlois & Garzarelli, 2008).

It has been noted in literature that there is no uniform definition of modularity (MacCormack, Rusnak, & Baldwin, 2006; Fixson & Clark, 2002; Fixson S. K., 2003; Bask, Lipponen, Rajahonka, & Tinnila, 2010). According to Campagnolo and Camuffo, "Modularity is an attribute of a complex system that advocates designing structures based on minimizing interdependence between modules and maximizing interdependence within them that can be mixed and matched in order to obtain new configurations without loss of the system's functionality or performance" (Campagnolo & Camuffo, 2010). Booch defined modularity as, "Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules" (Booch, 1994). Coupling is defined as how inter dependent two modules are and cohesion is defined as how single minded a module is (Yourdon & Constantine, 1979).

We recognize that it is difficult to come up with an absolute definition of modularity; on the other hand it is easy to conceptualize modularity. In this essay, we define modularity as a relative term. As an IS application is subdivided into more and more modules its modularity increases. Since, we are

only interested to investigate the effects of increase and decrease of our decision variables, it is not necessary to come up with an absolute definition of modularity. For a highly integral design (few modules and low modularity) cohesion of each module is low and coupling among the modules is also low. On the other hand, for highly modular design (many modules and high modularity) cohesion of each module is high, however coupling among modules also tend to be high. So we can infer that as modularity increases, coupling and cohesion also increases. Although operationalization and measurement of modularity is beyond the scope of this essay, we note that it is easy to measure effects of modularity indirectly through coupling and cohesion. Chidamber and Kemerer introduced specific metrics such as "Lack of Cohesion of Methods (LOCM)", "Coupling between Objects (CBO)" that could be used for measuring cohesion and coupling (Chidamber & Kemerer, 1994). Some application development tools such as Eclipse allows developers to calculate those metrics (Sourceforge.net). Ideally, it is best to have high cohesion and low coupling (Booch, 1994). We introduced the concept of modularity using the figures in the mausoleum of the first Chinese emperor Qin Shi Huang. Instead of just constructing head, arms, legs and torsos separately, the artists could have increased the number of modules by constructing fingers separately. In that case both cohesion of a module and coupling among the modules would have been higher. Intuitively we can sense that the product would have been less robust, as it would have been

harder to glue all the modules together. Hence neither too little modularity nor too much modularity is good. Hence, we conclude that there is an optimum modularity, which is neither too high nor too low. As high modularity corresponds to high cohesion and high coupling we can also assume that optimal modularity leads to high cohesion and low coupling.

2.3.3 Performance

Performance has been recognized as another important non-functional attribute of a software application (Devaraj, Kumar, Kavi, & Kanth, 2011). Also, performance and quality of service are two related constructs that are considered important attributes of telecommunication and networking services. A quick analysis of the Internet Service Providers (ISP) shows that ISPs charge higher price for products with higher performance as measured in upload and download speeds. We also observe the CPU manufacturers such as Intel and AMD advertise about the speeds of their CPUs. In the traditional IS area, performance is considered a runtime construct; hence during development of traditional IS applications performance did not receive much importance; it was assumed that performance issues could be addressed later (Balsamo, Di Marco, Inverardi, & Simeoni, 2004). However, in the data networking area during development of protocols, performance was considered an important attribute (Clark, 1982). In the area of scientific computing, considerable effort is made to develop software with higher performance (Diaz & Dutt, 1992). Also in the software development, there

are specific programming techniques that are used for improving performance of software under specific conditions. As an example data structure tree is used for data that does not change often to improve performance in searching. Hence performance should not be considered only a runtime construct and the role performance plays in software architecture during development should also be recognized. We recognize that by assuming that performance attribute of any IS application consists of two parts – architectural performance (s) and operational performance (o) so that the role of performance in the above contexts could be recognized. We define architectural performance (s) as the component of performance of an IS application that is due to design decisions and choices made during design and development of the IS application. A good example is a decision whether to use a specific data structure. We define operational performance as the component of performance of an IS application arising from the decisions made during run time of an IS application. Operational performance is the performance as observed by a user of a SaaS application (Transaction Processing Performance Council (TPC), 2012). So the operational performance depends on the decisions made during installation of IT infrastructure, such as type of hardware, amount of memory of the servers, they networking link that connects the servers etc.

For SaaS applications, the end users are not necessarily in the same enterprise where the application is hosted. Hence, the users of such applications could experience lower operational performance because of network delay, network attacks etc. A higher architectural performance could compensate for that. Also two similar SaaS products could be differentiated on the basis of performance, and a quick review of popular SaaS products show the importance SaaS providers give to the performance issue. Hence, we include architectural performance (s) as a decision variable in our model. However, we are only interested in relative changes in architectural performance and hence operationalization of architectural performance is beyond the scope of this essay. However, metrics such as TPC-W metrics (Transaction Processing Performance Council) has been used for measuring performance in the cloud computing area (Kossmann, Kraska, & Loesing, 2010).

2.4 Previous Research

A search on many popular databases such as "Web of Science", "ABI Inform" revealed a paucity of peer reviewed research papers on "Cloud Computing". As the focus of this paper is on SaaS, we have given more emphasis on SaaS literature. However, we have reviewed literature in other related areas of cloud computing also. In order to understand modularity and performance we have done an extensive review of those constructs in different areas.

Cloud computing in general and SaaS in particular have created a need for IS researchers to understand and use service science (Vaquero, Rodero-Merion, Caceres, & Lindner, 2009). Demirkan et al discussed importance of service orientation in IS applications and suggested some guidelines for developing and adopting IS applications based on service oriented architecture (Demirkan, Kauffman, Vayghan, Fill, Karagiannis, & Maglio, 2008). Bardhan et al confirmed that IT services were becoming more and more important in the IS area. They emphasized the importance of looking into service science for understanding the phenomenon and the need for joint research among different disciplines (Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010).

Recently, Susarla et al investigated on the suitability of high powered versus low powered incentives in the contract between a SaaS provider and a SaaS consumer. From the point of view of SaaS providers, it is more beneficial to have high powered incentive contract (as an example fixed price contract, giving the provider flexibility) instead of a low powered incentive contract (where the contract describes in detail different terms and conditions) between a SaaS application provider and a consumer for SaaS applications. It was observed that lack of flexibility in the contracts was one of the problems faced by Application Service Providers (ASP) (Susarla, Barua, & Whinston,

2009). They observed that modularity in SaaS application lowers the process specificity between consumers and producers and as a result of that makes it beneficial for SaaS providers to have high powered incentive contract with the consumers (Susarla, Barua, & Whinston, 2010).

Zhang and Seidman compared the subscription licensing model (similar to SaaS model), perpetual licensing model and a hybrid model that includes both types of licensing models for delivery of software. Although in some cases subscription licensing model was beneficial to software vendors, when the network effect is significant it was more profitable for vendors to provide both subscription licensing model and hybrid model (Zhang & Seidmann, 2010). Choudhary examined the impact of SaaS licensing scheme on software quality and he observed that SaaS licensing scheme leads to better software quality (Choudhary, 2007); because this licensing scheme encourages the producers to make more investment during software development and maintenance (Choudhary, 2007).

Demirkan et al studied different coordination strategies among different players in a supply chain of SaaS providers. They noted that two distinct roles have evolved for SaaS application providers – that of Application Service Providers (ASP) and of Application Infrastructure Providers (AIP). In order to provide SaaS, an ASP and an AIP can form a supply chain network. They

observed although an ASP and an AIP have different incentives, it is possible to create a coordination strategy with an incentive that will result in the same overall surplus as that could be achieved by a central planner (Demirkan, Cheng, & Bandyopadhyay, 2010). Benlian et al investigated drivers for adoption of SaaS applications. They found adoption of SaaS application depends on the type of specific application; however they found no relationship between SaaS adoption and the size of the adopting organizations. In other words SaaS vendors should not limit their marketing efforts based on the size of the consumer organizations (Benlian, Hess, & Buxman, 2009).

The concept of modularity in the area of management science has a long history. Modularity has been studied in three different areas – product design, production systems and organizations (Campagnolo & Camuffo, 2010). Parnas uncovered the importance of modularity in software architecture while discussing information hiding (Parnas, 1972). It was confirmed in many other studies that modular software is better software (Booch, 1991; Boehm & Sullivan, 2000; Cai, 2006; McConnell, 2000; Wasserman, 1996). For mainstream products, Joglekar and Rosenthal observed that use of modularity in software architecture improved outcomes of mainstream product which has added software components (Joglekar & Rosenthal, 2003), or in other words products with modular architecture are better products. Dewan et al investigated mass customization of product; they observed that

using mass customization, it was possible for a producer to offer different variations of a product at different prices leading to increased profit by making the product attractive to a more diverse group of consumers (Dewan, Jing, & Seidmann, 2003). Kumar observed that modularity in product design enables a producer to offer mass customized product (Kumar, 2004). Hence, it is widely accepted that increase in modularity leads to increase in product flexibility (Schilling, 2000).

Modularity is also one of the driving forces for refactoring and restructuring of software (Mens & Tourwe, 2004). MacCormack et al observed modularity in design led to more flexibility in changing products and that increased agility (MacCormack, Verganti, & Iansiti, 2001). It was also observed that maintenance cost was proportional to size of the modules amongst others; it was more expensive to maintain a non-modular product (Banker, Datar, Kemerer, & Zweig, 1993) or in other words the cost for maintaining a modular product is less. However, many researchers in different fields have observed that modularity in product architecture leads to higher product complexity (Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010; Baldwin & Clark, 2000). Pekkarinen and Ulkuniemi studied how modularity could be introduced during development of business services. They developed a modular services platform where they identified four distinct areas of a typical business process - service, process, organisational and customer

interface. They showed it was possible to introduce modularity in all the four areas and improve the overall business process (Pekkarinen & Ulkuniemi, 2008).

The literature on architectural performance of software product is not extensive. Jain and Kannan showed that in a software service environment, price is related to performance and producers charge higher price for higher performing product (Jain & Kannan, 2002). Hosanger et al studied the role of performance on a specific IS service namely cache service in the context of consumer vendor relationship. They showed that it was possible for vendors to charge for a premium service even when a best effort free service was available to consumers (Hosangar, Krishnan, Chuang, & Choudhary, 2005) . Hanmer and Letourneau described some of the best practices for developing a high performing product (Hanmer & Letourneau, 2003).

The importance of service in IS applications is evident from another perspective. Service Oriented Architecture (SOA) has become a very popular IS architecture; SOA could be defined as an architectural style where each component of an IS is perceived as a service. SOA architecture enables applications built using modular structure to work together. A logical next step is to extend SOA outside of the enterprise to the cloud (Linthicum, 2009). Hence, it has been suggested that for building cloud computing

enabled IS applications, SOA is the most appropriate architecture (Demirkan H. , 2008).

2.5 Model Formulation:

We use theoretical (analytical) modeling technique for our model formulation. In this technique researchers build theoretical models which consist of appropriate variables for the phenomenon which is being modeled as well as realistic assumptions among the variables. A typical assumption could be an assumption of profit maximization. They do theoretical experiments using the model by examining effects of change of some of the variables. From the results of the experiments, propositions are developed and managerial implications of the phenomenon are uncovered from the propositions (Moorthy, 1993). Theoretical modeling has been an effective tool for uncovering many useful insights into complex business phenomena (Raju, 1995).

We first model our demand function. We assume a monopolist vendor and we consider a fixed period during which the SaaS application is being offered. We model the effects of modularity and performance in SaaS architecture on demand. Consistent with the literature, we assume that modularity and performance have positive effects on the demand. Bakos and Brynjolfsson observed that especially in the case of Information System products, bundling

leads to higher profit for producers (Bakos & Brynjolfsson, 1999); modularity in product architecture leads to ease in bundling. In other words, modularity in product architecture will enable a SaaS provider to create and customize their services faster for the consumers (Sambamurthy, Bharadwaj, & Grover, 2003). Joglekar and Rosenthal observed that use of modularity in software architecture improved outcomes of mainstream product which has added software components (Joglekar & Rosenthal, 2003). Modularity would also support mass customization strategy which allows a producer to offer their product to a more diverse group of customers (Dewan, Jing, & Seidmann, 2003). Therefore, an increase in modularity of a product will lead to increase in demand if everything else remains the same.

One of the most successful cloud computing application providers is Salesforce.com. Salesforce.com has seen a steady increase in their sales and customer base that has been attributed to the flexibility of its Application Programming Interface (API) Force.com that allows users to develop their own applications. According to Salesforce.com website, "Force.com comes with 60 predefined components that can be assembled with minimal coding in building-block fashion. Some of these components implement common Salesforce interface elements and others make new features available, such as AJAX-based partial page refreshes" (Salesforce.com). In other words,

modularity in their products makes it easy for customers to develop their own application leading to Salesforce.com's success.

Another success story in the area of cloud computing is popularity of Amazon Web Services. According to Adam Selipsky, vice president of Product Management and Developer Relations, Amazon Web Services, "in making its capabilities accessible to outside developers, Amazon broke its process into many modular services. This modularity has allowed Amazon to extend its business all the way to providing a complete online retailing environment for Target.com, Marks & Spencer, and others." Hence, it follows that introduction of modularity in Amazon's processes led to increase in number of their customers.

According to Catalyst Resources a SaaS consulting firm, modular design in SaaS leads to order of magnitude profit increase for SaaS developers. They gave an example, where a client of theirs opted for a non-modular design resulting in loss of profit – "A company we work with had a very compelling piece of software for transportation asset management. However, all functionality in the application was bolted together. All configuration was in one area, all reports were in one area, all routing & logistics were in one area, etc. This meant the company was limited to selling their SaaS offerings as a single product at a single subscription price. Ideally with SaaS however,

you want is to be able to break your functionality into pieces that can be sold separately. The modular pieces become separate profit streams that sum to more than the profit from a single monolithic SaaS.”

On the effect of performance and demand level it has been argued that higher performance will mean higher demand if everything else remains the same. Hosangar et al confirmed that indirectly in a recent study (Hosangar, Krishnan, Chuang, & Choudhary, 2005). Kossman et al compared the performances of different cloud computing services such as Amazon Web Services (AWS), Google’s AppEngine, Microsoft’s Azure with respect to the price of those services. They used TPC-W metrics of the Transaction Processing Performance Council (Transaction Processing Performance Council). The performance was measured in WIPS (Web Interactions per Second). They observed that providers charge higher prices for higher performing product (Kossmann, Kraska, & Loesing, 2010). From the fundamental laws of economics we note that, lower prices lead to higher demands. Hence, we argue if everything else remains same higher architectural performance will lead to higher overall performance and that will lead to higher demand if the price remains unchanged as that would be effectively lowering of prices.

Some of the examples from industry as discussed above clearly indicate that modularity in product leads to increase in demand and lack of modularity in product leads to decrease in demand.

The literature predicts similar trend for performance. We assume a linear demand function and we include the sensitivity in demand from modularity and architectural performance as discussed earlier. Although linear demand function has some limitations, linear demand function is widely used in the literature (Barua, Kriebel, & Mukhopadhyay, 1991; Choudhary, 2007). We also assume both demands and price to be amortized over the lifetime of the product. If a consumer subscribes to the service throughout its lifetime, the price is the total amount the consumer will pay and correspondingly demand will be one unit. Hence both demand and price could be fractions. This is how a service is different from product. A consumer can only purchase or not purchase a product. On the other hand, a consumer can subscribe to a service for a limited period of time and not throughout the lifetime of the service. In that case, a consumer will pay less than the full price and demand will be recognized as less than 1.

Thus, demand can be modeled as:

$$d = \alpha - \beta p + \gamma m + \delta s \quad (1)$$

where p is price of the SaaS application amortized over its lifetime,

m is the modularity level of the SaaS application,

and s is the performance level of the SaaS application.

α is primary demand due to functional attributes of the SaaS application and other non-functional attributes such as quality (except modularity and performance), brand image, general economic fact that are outside the scope of this paper.

β represents price sensitivity of the demand,

γ represents increase in demand from increase in modularity,

and δ represents increase in demand from increase in performance.

α , β , γ , and δ are assumed to be greater than zero.

Our next step is to formulate the cost function. A provider of SaaS application incurs three different types of costs. First, there is a fixed cost which involves the cost of developing the product as well as costs for setting up the necessary IT infrastructure so that the SaaS application could be offered to the customers. Second, the providers also incur a marginal cost per service because in order to provide larger number of services to larger number of customers, it is necessary to have a larger IT infrastructure. The marginal cost also consists of two parts. There could be a onetime cost of purchasing the infrastructure and a variable personnel cost of maintaining the

infrastructure and providing the service. Alternatively, the infrastructure could be rented from an IaaS provider and in that case the marginal cost will only consist of a variable component. Third, there will be a cost for maintaining the SaaS application. In our model, we include both maintenance cost and marginal cost per product as amortized over the lifetime of the product.

Prior research has shown that modularity in product architecture leads to higher product development complexity (Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010). Hence we can infer production of modular software will require more production cost for vendors (i.e. higher upfront (fixed) cost) (Bush, Tiwana, & Rai, 2010). We assumed that fixed cost (C_1) arising from increased modularity and performance to be a quadratic function of modularity (m) and performance (s) respectively. This is in line with the standard practice in the IS literature; fixed costs incurred to improve quality of a product is a convex function of the slope of product improvement curve (Choudhary, 2007). Therefore C_1 can be expressed as:

$$C_1 = A_1 + C m^2 + D s^2 \quad (2)$$

where A_1 is the fixed cost arising from factors other than modularity and performance, C is the parameter related to modularity during design and development of the application, and D is the parameter related to performance during design and development of the application.

Modularity in design also leads to better flexibility in changing products leading to agility (MacCormack, Verganti, & Iansiti, 2001). It was more expensive to maintain a non-modular product compared to a modular product (Banker, Datar, Kemerer, & Zweig, 1993; Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010). We treat maintenance cost (C_2) as amortized over the lifetime of the product. Hence, C_2 can be expressed as:

$$C_2 = A_2 - B m \quad (3)$$

where A_2 is the amortized maintenance cost over the lifetime of the product and

B , is the parameter related to modularity showing the saving in maintenance cost arising from modular design also amortized over the lifetime of the product.

Unlike a traditional software vendor, a SaaS application provider will also incur marginal cost for providing services. This marginal cost (C_3) will include both the cost for setting up infrastructure such as hardware, software as well as the amortized cost for providing the service. This cost will be proportional to demand. Hence, C_3 can be expressed as:

$$C_3 = Z d \quad (4)$$

where, d is the amortized demand of the service , and Z is the marginal cost per application amortized over the total lifetime of the service.

Adding (2), (3), and (4), our total cost function can be expressed as:

$$\theta = A - B m + C m^2 + D s^2 + Z d \quad (5)$$

where $A = A_1 + A_2$

Profit for a software producer (π) could be represented as:

$$\pi = d p - \theta$$

Using (1) and (5), the above profit can be rewritten as:

$$\pi = (\alpha - \beta p + \gamma m + \delta s)(p - Z) - (A - B m + C m^2 + D s^2) \quad (6)$$

Our objectives are to find optimal values of price (p), modularity (m) and software performance (s) that will maximize the above profit function.

Boundary Condition:

We assume the boundary condition that *Primary demand of a product a is greater than product of marginal cost Z and price sensitivity β or $\alpha > \beta Z$*

In any IS application, it is fair to assume that functional attributes are much more important than non-functional attributes. Although non-functional attributes are important in this case, it is unlikely that importance of non-functional attributes such as modularity and performance will be more than those of functional attributes and other non-functional attributes (except modularity and performance). It is unlikely that a producer will produce a

product based on only modularity and performance and without any regard to its functionality. We consider a case where both performance sensitivity (δ) and modularity sensitivity (γ) are close to zero or in other words, the customers do not care about the product's modularity and performance. Following equation (1), we can rewrite the demand function as

$$d = \alpha - \beta p$$

In the above case, it is unlikely that a producer will produce product that does not have a projected positive demand. Hence, we assume that in this simplistic situation also demand should be positive. Hence, we obtain

$$\alpha > \beta p$$

As Z is the unit marginal cost, it is fair to assume that p must always be greater than Z , since otherwise it does not make sense for a producer to produce any product. As we are modeling the situation during design and production phase, we exclude the possibility of a fire sale. Hence it follows

$$\alpha > Z\beta \tag{7}$$

We shall assume the above boundary condition in our model.

2.6 Results:

In this section, we present our results. We consider two cases. In the first case we assume no relationship among the decision variables. In the second case, we assume a relationship between modularity (m) and architectural performance (s) where increase of one leads to decrease of the other.

2.6.1 Case 1: Modularity (m) and Architectural Performance (s) are not related

In this case, we assume modularity in software architecture and the architectural performance are independent of each other. In order to find optimal values for our decision variables that will maximize the profit function, we differentiate profit (Equation 6) partially with respect to p , m , and s , set them equal to zero, and express them as p^* , m^* , and s^* . After making some simplifications, we obtain the following expressions.

$$p^* = \frac{\alpha + Z\beta + \gamma m + \delta s}{2\beta} \quad (8)$$

$$m^* = \frac{B + \gamma(p - Z)}{2C} \quad (9)$$

$$s^* = \frac{\delta(p - Z)}{2D} \quad (10)$$

Solving above equations by substituting p^* , m^* and s^* for p , m and s respectively, we obtain the optimal values of decision variables as expressed below.

$$p^* = \frac{2CD\alpha + BD\gamma + Z(2CD\beta - D\gamma^2 - C\delta^2)}{4CD\beta - D\gamma^2 - C\delta^2} = \frac{2CD(\alpha - Z\beta) + BD\gamma}{4CD\beta - D\gamma^2 - C\delta^2} + Z \quad (11)$$

$$m^* = \frac{4B D \beta + 2 D \alpha \gamma - 2 D Z \beta \gamma - B \delta^2}{2 (4 C D \beta - D \gamma^2 - C \delta^2)} \quad (12)$$

$$s^* = \frac{(2 C \alpha + B \gamma - 2 C Z \beta) \delta}{2 (4 C D \beta - D \gamma^2 - C \delta^2)} \quad (13)$$

Substituting (8), (9), and (10) to (1) and (6), optimal demand and profit can be expressed as:

$$d^* = \frac{D \beta (2 C (\alpha - Z \beta) + B \gamma - 2 C)}{(4 C D \beta - D \gamma^2 - C \delta^2)} \quad (14)$$

$$\pi^* = \frac{4 D (\alpha - Z \beta) (B \gamma + C (\alpha - Z \beta)) + B^2 (4 D \beta - \delta^2)}{4 (4 C D \beta - D \gamma^2 - C \delta^2)} - A \quad (15)$$

Lemma 1:

All the decision variables as well as the optimal demand and profit have positive values when $4 C D \beta > (D \gamma^2 + C \delta^2)$.

Proof:

The above condition is derived from the hessian matrix. The hessian matrix is the second-order partial derivative of the profit function with respect to the variables p, m and s in the present case.

$$H = \begin{pmatrix} -2\beta & \gamma & \delta \\ \gamma & -2C & 0 \\ \delta & 0 & -2D \end{pmatrix}$$

To ensure that profit has a local maximum the determinants of the Hessian matrix need to be negative semi definite. A matrix is negative semi definite when its leading principal minors of different orders alternate in sign, starting with negative for the first leading principal minor (Winston, 1993). Hence the principal minor of order 3 has to be negative. The only principal minor of order three is the determinant of the matrix itself which is $2(-4 C D \beta + D \gamma^2 + C \delta^2)$, and that leads to the following condition.

$$4 C D \beta > (D \gamma^2 + C \delta^2) \quad (16)$$

We note that first principal minor of second order is $(4C\beta - \gamma^2)$. However, it is positive when Equation (16) is assumed. Using Equation (16) as well as the boundary condition, the numerators and denominators in Equations (11), (12), (13), (14), and (15) are all positives. ■

Next, we study the effect of various sensitivity parameters to our decision variables as well as to our optimal profit.

Proposition 1:

In a market where customer demand is such that it is more sensitive to modularity (i.e. higher γ), the vendors will be able to charge a higher price.

Proof:

Taking the partial derivative of optimal p^* with respect to γ , we obtain

$$\frac{\partial p^*}{\partial \gamma} = \frac{D (B (4 C D \beta + D \gamma^2 - C \delta^2) + 4 C D (\alpha - Z \beta))}{(4 C D \beta - D \gamma^2 - C \delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. From Lemma 1 and boundary condition of equation (7), we find $4 D \beta - \delta^2 > 0$ and $(\alpha - Z \beta) > 0$ respectively. Hence the numerator is always positive.

Therefore, $\frac{\partial p^*}{\partial \gamma} > 0$.

■

Hence, we conclude that if all the other parameters remain the same, increase in customer preference for modularity will enable vendors to charge a higher price.

Proposition 2:

In a market where customer demand is such that it is more sensitive to modularity (i.e. higher γ), the vendors will offer a product that is more modular.

Proof:

Taking the partial derivative of optimal m^* with respect to γ we obtain

$$\frac{\partial m^*}{\partial \gamma} = \frac{D ((4 D \beta - \delta^2)(C (\alpha - Z \beta) + B\gamma) + D\gamma^2(\alpha - Z\beta))}{(4 C D \beta - D \gamma^2 - C \delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. From Lemma 1, since $4 D \beta - \delta^2 > \frac{D\gamma^2}{C}$, the numerator can be rewritten to

$$D\left(\frac{D\gamma^2}{C}\right)(C (\alpha - Z \beta) + B\gamma) + D\gamma^2(\alpha - Z \beta).$$

From Lemma 1 we find $(\alpha - Z \beta)$ is positive; hence the numerator is always positive.

Therefore,

$$\frac{\partial m^*}{\partial \gamma} > 0 \quad \blacksquare$$

Hence, we can conclude that if all the other parameters remain same, increase in customer preference for modularity will lead to vendors making the SaaS applications more modular. However, we also need to remember that the optimal modularity level depends on many other factors such as the cost for introducing modularity; hence the implication of this proposition will be clearer as we develop other propositions. We show the above results graphically in the next page.

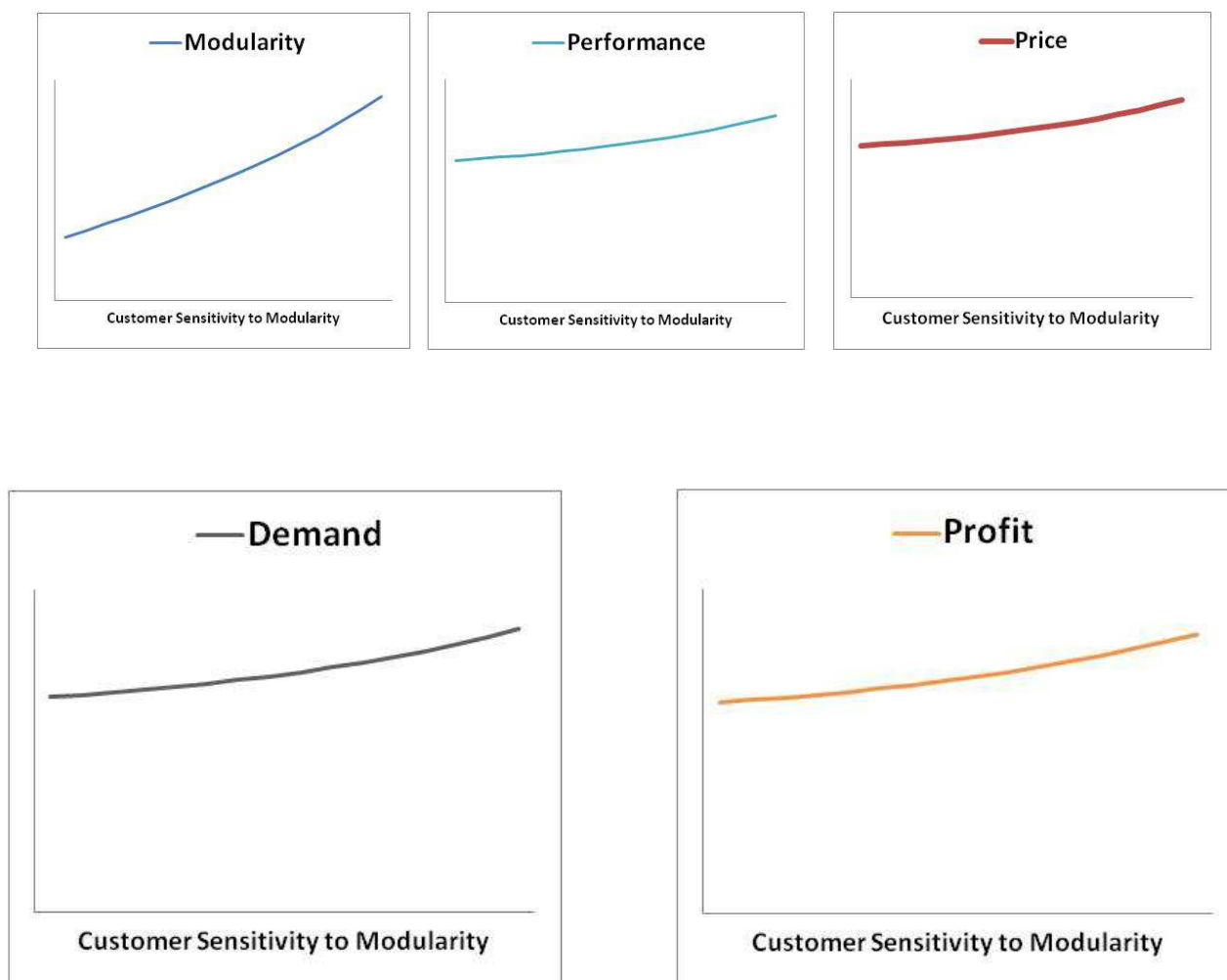


Figure 2.1: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to customer sensitivity to modularity in the IS application architecture

Our numerical results confirm our analytical results obtained in propositions and lemma 1.

We also find that propositions (1) and (2) are consistent. Increase in customer preference for modularity will lead to vendors making the SaaS applications more modular and as a result of that they will be able to charge a higher price.

Proposition 3:

As the cost of introducing modularity in the product increases, a vendor will be required to lower the optimal price.

Proof:

Taking the partial derivative of optimal p^* with respect to C we obtain

$$\frac{\partial p^*}{\partial C} = - \frac{D\gamma(4BD\beta + 2D\alpha\gamma - B\delta^2 - 2DZ\beta\gamma)}{(4CD\beta - D\gamma^2 - C\delta^2)^2}$$

The denominator of the above equation is always positive as it is a square of an expression. Next, we can rewrite the numerator to:

$$\begin{aligned} &= -D\gamma(4BD\beta - B\delta^2 + 2D\alpha\gamma - 2DZ\beta\gamma) \\ &= -D\gamma(B(4D\beta - \delta^2) + 2D\gamma(\alpha - \beta Z)) \end{aligned}$$

Using Lemma 1 and the boundary condition, it can be shown that the numerator will be negative. Therefore, $\frac{\partial p^*}{\partial C} < 0$. ■

Proposition 4:

As the cost for introducing modularity in the product increases, a vendor will be required to lower the optimal modularity level as well as the optimal performance level.

Proof:

By taking the partial derivative of optimal m^* with respect to C we obtain

$$\begin{aligned} \frac{\partial m^*}{\partial C} &= - \frac{(4D\beta - \delta^2)(2D(\alpha - Z\beta)\gamma + B(4D\beta - \delta^2))}{2(4CD\beta - D\gamma^2 - C\delta^2)^2} \\ &= - \frac{(4D\beta - \delta^2) m^*}{(4CD\beta - D\gamma^2 - C\delta^2)} \end{aligned}$$

Using Lemma 1, we observe both numerator $(4D\beta - \delta^2) m^*$ and denominator $(4CD\beta - D\gamma^2 - C\delta^2)$ are positive. Hence, $\frac{\partial m^*}{\partial C}$ is negative of a positive quantity.

Therefore, $\frac{\partial m^*}{\partial C} < 0$.

By taking the partial derivative of optimal s^* with respect to C , we obtain

$$\frac{\partial s^*}{\partial C} = - \frac{\gamma\delta(2D(\alpha - Z\beta)\gamma + B(4D\beta - \delta^2))}{2(4CD\beta - D\gamma^2 - C\delta^2)^2} = - \frac{\gamma\delta m^*}{(4CD\beta - D\gamma^2 - C\delta^2)}$$

Using Lemma 1, we observe that numerator $\gamma\delta m^*$ and denominator $(4CD\beta - D\gamma^2 - C\delta^2)$ are positive. Hence, $\frac{\partial s^*}{\partial C}$ is negative of a positive quantity.

Therefore,

$$\frac{\partial s^*}{\partial C} < 0. \quad \blacksquare$$

Propositions (3) and (4) are very interesting. As the cost of introducing modularity increases, the price as well as the optimal modularity and performance levels decrease. Actually, proposition (4) explains proposition (3). If the cost of introducing modularity becomes higher then ceteris paribus, a vendor will not benefit by making the product more modular; instead the vendor will produce a product with lower modularity and performance levels. As a result of that a vendor will be required to lower price at the same time. Following figures show above results graphically.

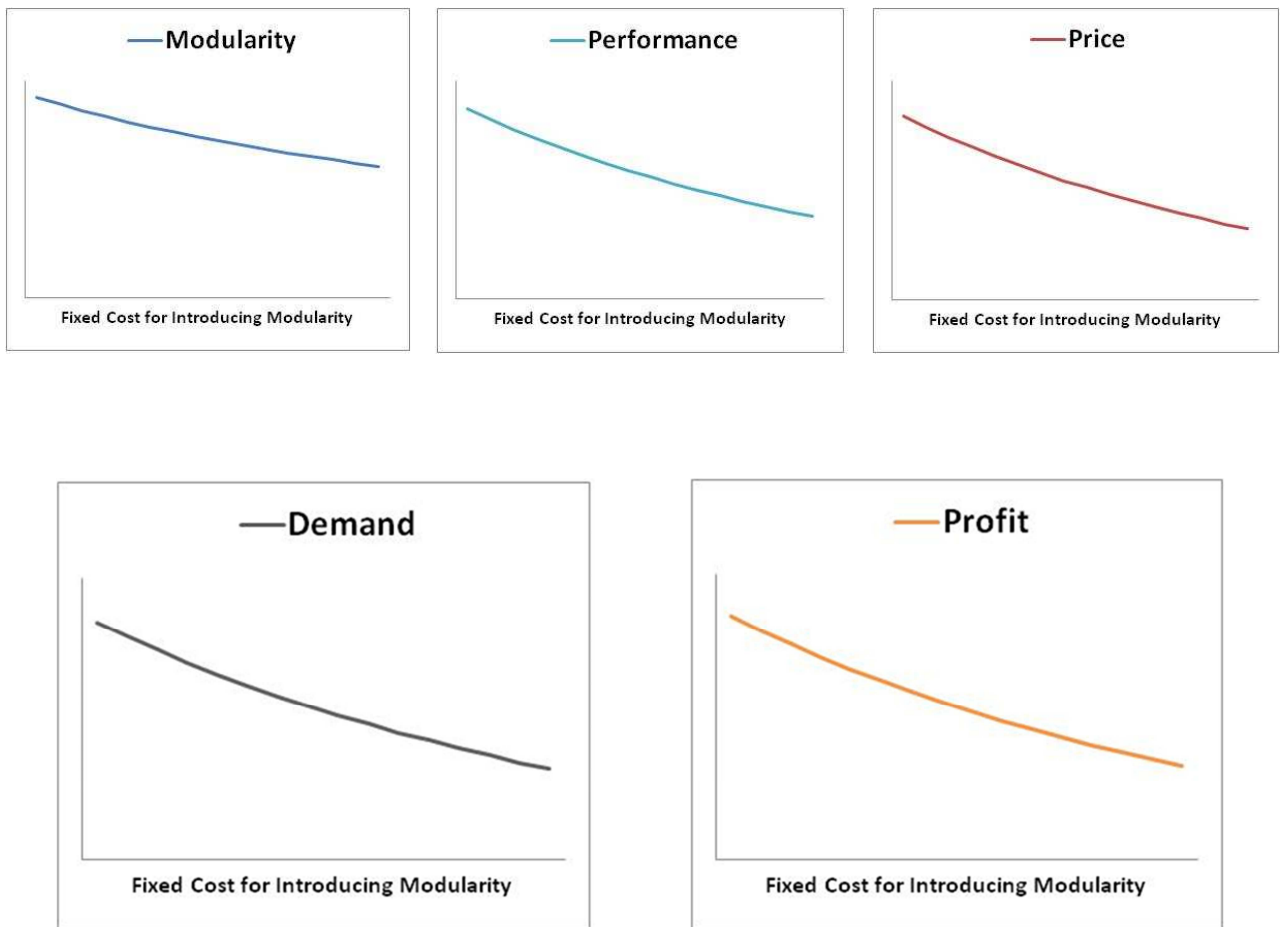


Figure 2.2: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to change in fixed cost for introducing modularity into the system

Proposition 5:

As the marginal cost Z increases the producers need to

(a) increase optimal price if $\beta > \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$

(b) decrease optimal price if $\beta < \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$

(c) keep optimal price unchanged if $\beta = \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$

Proof:

By taking the partial derivative of optimal p^* with respect to Z , we obtain

$$\frac{\partial p^*}{\partial Z} = \frac{2CD\beta - D\gamma^2 - C\delta^2}{4CD\beta - D\gamma^2 - C\delta^2}$$

From Lemma 1, we conclude that denominator is always positive. Depending on the numerator, $\frac{\partial p^*}{\partial Z}$ becomes positive, negative or zero.

$$\text{Numerator} = 2C D \left(\beta - \frac{\gamma^2}{2C} - \frac{\delta^2}{2D} \right)$$

If $\beta > \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$, then numerator is positive.

If $\beta < \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$, then numerator is negative.

If $\beta = \frac{\gamma^2}{2C} + \frac{\delta^2}{2D}$, then numerator is zero.

■

Proposition 5 has a very important managerial implication. It can be shown that with increase in marginal cost, all the decision variables except optimal price decrease. Depending on different conditions as indicated above between sensitivities and fixed costs, optimal price may remain unchanged, increase or decrease.

2.6.2 Case 2: Modularity (m) and Performance (s) are related

In the literature it has been observed that increase in modularity leads to decrease in performance (Ulrich, 1995). Lau Antonio et al found that although modularity in product design is considered a key enabler of product success, modularity does not necessarily improve all the attributes of a product (Lau-Antonio, Yam, & Tang, 2007). While discussing modularity and performance in protocol implementation Clark observed in Request For Comments (RFC) 817 published by the Internet Engineering Task Force (IETF) that "modularity is one of the chief villains in attempting to obtain good performance, so that the designer is faced with a delicate and inevitable tradeoff between good structure and good performance" (Clark, 1982). We modify our above model to include an inverse relationship between performance, and modularity; an increase in modularity leads to decrease in performance and vice versa.

Hence we assume,

$$s = X - Y m \quad (17)$$

where X and Y are parameters. X shows the maximum performance level and Y shows the ratio of change in s to change in m . It also follows that maximum value of m is X/Y (when s is zero). We also note that neither X nor Y is fixed.

Using equation (1) and eliminating s using equation (17) we can rewrite our linear demand function as

$$d = \alpha - \beta p + (\gamma - \delta Y) m + X\delta \quad (18)$$

Using (6) and (17), we can reformulate our profit function as

$$\begin{aligned} \pi &= (\alpha - \beta p + \gamma m + \delta s)(p - Z) - (A - B m + C m^2 + D(X - Y m)^2) \\ &= (\alpha - \beta p + (\gamma - \delta Y) m + X)(p - Z) - (A - (B + 2DXY) m + (C + DY^2)m^2 + DX^2) \end{aligned} \quad (19)$$

We differentiate profit partially with respect to p and m , set them equal to zero, and solve for p and m . After making some simplifications, we obtain the following expressions for optimal price and modularity.

$$p_{ms}^* = \frac{2(\alpha + Z\beta)(C + DY^2) - Z(\gamma - Y\delta)^2 + 2X(C\delta + DY\gamma) + B(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

$$p_{ms}^* = \frac{2(\alpha - Z\beta)(C + DY^2) + 2X(C\delta + DY\gamma) + B(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} + Z \quad (20)$$

$$m_{ms}^* = \frac{2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \quad (21)$$

Using equations (17) and (21), we can express optimal performance as

$$s_{ms}^* = X - \frac{Y(2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta))}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

$$s_{ms}^* = \frac{4\beta XC - 2\beta YB - (\gamma - Y\delta)(Y(\alpha - Z\beta) + X\gamma)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \quad (22)$$

Substituting (20), (21), and (22) to (18) and (19), our optimal demand and profit can be expressed as:

$$d_{ms}^* = \frac{\beta (2 (\alpha - Z\beta)(C + DY^2) + 2X(Y\gamma D + C) + B(\gamma - Y\delta))}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} \quad (23)$$

$$\pi_{ms}^* =$$

$$\frac{B^2\beta + A(\gamma - Y\delta)^2 + B(\alpha - Z\beta + X\delta)(\gamma - Y\delta) + D((Y(\alpha - Z\beta) + X\gamma)^2 - 4\beta Y(A\gamma - XB)) + C((\alpha - Z\beta + X\delta)^2 - 4\beta(A + D X^2))}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

$$= \frac{B^2\beta + B(\alpha - Z\beta + X\delta)(\gamma - Y\delta) + D((Y(\alpha - Z\beta) + X\gamma)^2 + 4\beta X Y B) + C((\alpha - Z\beta + X\delta)^2 - 4\beta D X^2)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} - A \quad (24)$$

Boundary Conditions:

We make the following assumptions regarding this model.

1. As in case 1, we assume equation (7) or $\alpha > Z\beta$
2. We observe from equation (18), that the demand function has $(\gamma - \delta Y)$ as the effective sensitivity for modularity. It is fair to assume that sensitivity for modularity is never negative, or

$$(\gamma \geq \delta Y) \quad (25)$$

Lemma 2:

All the decision variables as well as the optimal demand and profit have positive values when

$$4\beta(C + D Y^2) > (\gamma - Y\delta)^2$$

Proof:

The Hessian matrix is

$$H = \begin{pmatrix} -2\beta & \gamma - Y\delta \\ \gamma - Y\delta & -2C - 2DY^2 \end{pmatrix}$$

To ensure that profit has a local maximum the Hessian matrix needs to be negative semi definite. A matrix is negative semi definite when its leading principal minors of different orders alternate in sign starting with negative (Winston, 1993). Hence the leading principal minor of order 2 needs to be positive. Hence, we obtain

$$4\beta(C + D Y^2) > (\gamma - Y\delta)^2 \quad (26)$$

Using Equation (26) as well as above boundary conditions, the numerators and denominators in Equations (20), (21), (23), and (24) are all positives.

We also assumed that maximum value of m is X/Y .

Hence,

$$m_{ms}^* = \frac{2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)}{4\beta(C + DY^2) - (\gamma - Y\delta)^2} < \frac{X}{Y}$$

From the above inequality, we obtain the following expressions

$$X(4\beta(C + DY^2) - (\gamma - Y\delta)^2) > Y(2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta))$$

$$4\beta X C > 2B Y \beta + (Y(\alpha - Z\beta) + X\gamma)(\gamma - Y\delta)$$

Above inequality shows, that the numerator of equation (22) is positive.

Hence, s_{ms}^* is also positive.

■

Proposition 6:

In a market where customer demand is such that it is more sensitive to modularity (i.e. higher γ), the vendors will be able to charge a higher price, by increasing modularity in the IS application architecture. It will also lead to higher demand and higher profit for the producers.

Proof:

In order to examine the variation in the optimal decision variables with respect to γ we differentiate p_{ms}^* , m_{ms}^* , d_{ms}^* and π_{ms}^* with respect to γ . We obtain

$$\frac{\partial p_{ms}^*}{\partial \gamma} = \frac{4C(2DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)) + B(4C\beta + 4DY^2\beta + (\gamma - Y\delta)^2) + 2DY(4DXY^2\beta + (\gamma - Y\delta)(X\gamma + Y(2\alpha - 2Z\beta + X\delta)))}{(4\beta(C + DY^2) - (\gamma - Y\delta)^2)^2}$$

$$\frac{\partial m_{ms}^*}{\partial \gamma} = \frac{(4C(2DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta)) + B(4C\beta + 4DY^2\beta + (\gamma - Y\delta)^2) + 2DY(4DXY^2\beta + (\gamma - Y\delta)(X\gamma + Y(2\alpha - 2Z\beta + X\delta))))}{(4\beta(C + DY^2) - (\gamma - Y\delta)^2)^2}$$

$$\frac{\partial d_{ms}^*}{\partial \gamma} = \frac{\beta(2(\gamma - Y\delta)(2DY(Y\alpha - YZ\beta + X\gamma) + 2C(\alpha - Z\beta + X\delta) + B(\gamma - Y\delta)) + (B + 2DXY)(4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2))}{(4\beta(C + DY^2) - (\gamma - Y\delta)^2)^2}$$

$$\frac{\partial \pi_{ms}^*}{\partial \gamma} = \frac{(2DY(Y\alpha - YZ\beta + X\gamma) + 2C(\alpha - Z\beta + X\delta) + B(\gamma - Y\delta)) * m_{ms}^*}{4\beta(C + DY^2) - (\gamma - Y\delta)^2}$$

The denominator is always positive as it is a square of an expression. By carefully examining the numerators in all the above four cases, we find that the only way the numerator could be negative in each case if $\alpha < Z\beta$ or $\gamma < Y\delta$ or both. According to our boundary conditions, we have $\alpha > Z\beta$ and $\gamma > Y\delta$. Hence, we conclude $\frac{\partial p_{ms}^*}{\partial \gamma} > 0$, $\frac{\partial m_{ms}^*}{\partial \gamma} > 0$, $\frac{\partial d_{ms}^*}{\partial \gamma} > 0$, and $\frac{\partial \pi_{ms}^*}{\partial \gamma} > 0$.

■

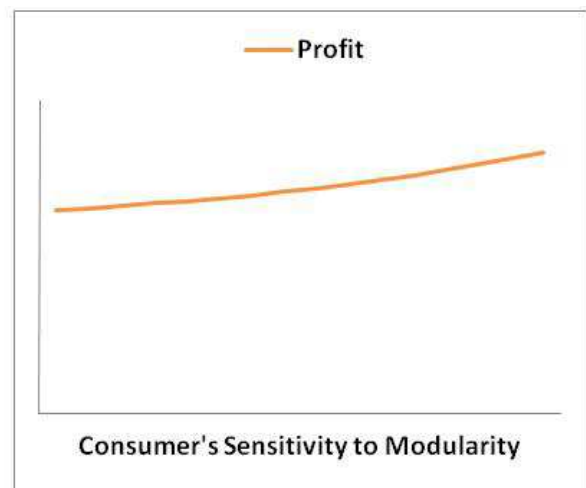
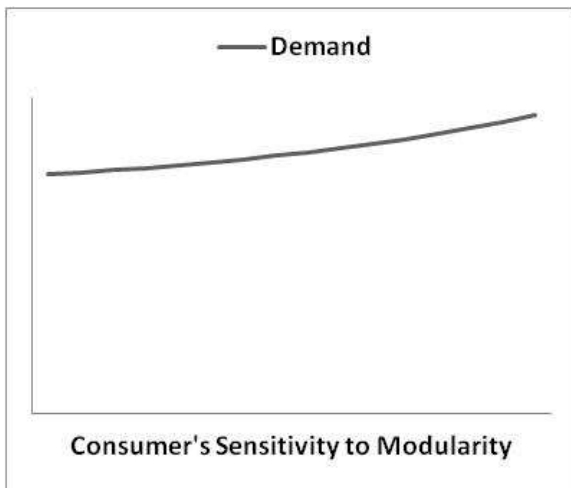
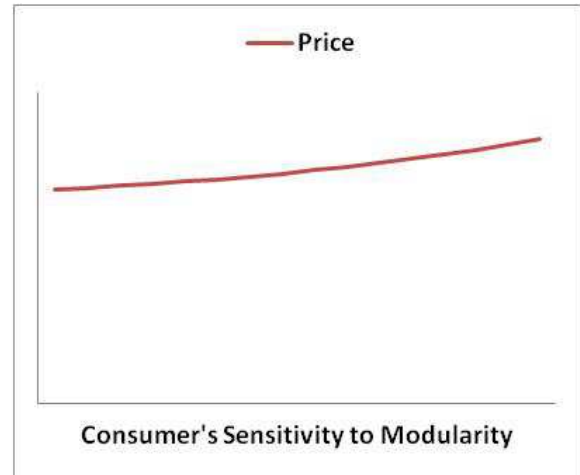
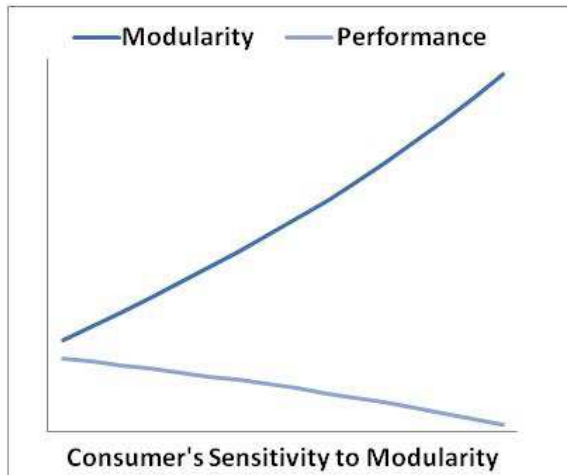


Figure 2.3: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to change in sensitivity to modularity.

Proposition 7:

When modularity and architectural performance are related to each other, the producer's profit is maximum with respect to both X and Y when

$$X = \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}$$

In the above case, all the decision variables are equal to their values as in the case where modularity and performance are not related.

Proof:

By taking the partial derivative of optimal π_{ms}^* with respect to X , we obtain

$$\begin{aligned} \frac{\partial(\pi_{ms}^*)}{\partial X} &= \frac{2(\alpha - Z\beta)(YD\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta)) + 2X(C\delta^2 + D\gamma^2 - 4CD\beta)}{4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2} \end{aligned}$$

By taking the partial derivative of optimal π_{ms}^* with respect to Y , we obtain

$$\begin{aligned} \frac{\partial(\pi_{ms}^*)}{\partial Y} &= - \frac{(2B\beta + 4DXY\beta + (\alpha - Z\beta + X\delta)(\gamma - Y\delta))(2(\alpha - Z\beta)(YD\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta)) + 2X(C\delta^2 + D\gamma^2 - 4CD\beta))}{(4C\beta + 4DY^2\beta - (\gamma - Y\delta)^2)^2} \end{aligned}$$

We observe,

$$\frac{\partial(\pi_{ms}^*)}{\partial Y} = - (m_{ms}^*) \left(\frac{\partial(\pi_{ms}^*)}{\partial X} \right)$$

We note, as m_{ms}^* is always positive, hence both $\frac{\partial(\pi_{ms}^*)}{\partial Y}$ and $\frac{\partial(\pi_{ms}^*)}{\partial X}$ always have

opposite signs and if one is zero then the other is zero too.

We observe using Lemma 2, denominator of $\frac{\partial(\pi_{ms}^*)}{\partial X}$ is always positive.

However, the numerator could be positive, negative or zero depending on the relationship between X and other parameters.

We observe, when

$$X < \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)},$$

$$\frac{\partial(\pi_{ms}^*)}{\partial X} > 0$$

That means π_{ms}^* will increase as X increases.

However, when

$$X > \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}, \quad \frac{\partial(\pi_{ms}^*)}{\partial X} < 0$$

Which means π_{ms}^* will decrease as X increases.

Hence, π_{ms}^* will be maximum with respect to X and Y, when

$$X = \frac{2(\alpha - Z\beta)(DY\gamma + C\delta) + B(4DY\beta + \delta(\gamma - Y\delta))}{2(4CD\beta - D\gamma^2 - C\delta^2)}$$

We also find from equations 20-24, for the above value of X, the optimal decision variables become

$$p_{ms}^* = \frac{2CD(\alpha - Z\beta) + BD\gamma}{4CD\beta - D\gamma^2 - C\delta^2} - Z = p^*$$

$$m_{ms}^* = \frac{4BD\beta + 2D\alpha\gamma - 2DZ\beta\gamma - B\delta^2}{2(4CD\beta - D\gamma^2 - C\delta^2)} = m^*$$

$$s_{ms}^* = \frac{(2C\alpha + B\gamma - 2CZ\beta)\delta}{2(4CD\beta - D\gamma^2 - C\delta^2)} = s^*$$

$$d_{ms}^* = \frac{D\beta(2C(\alpha-Z\beta)+B\gamma-2C)}{(4CD\beta-D\gamma^2-C\delta^2)} = d^*$$

$$\pi_{ms}^* = \frac{4D(\alpha-Z\beta)(B\gamma+C(\alpha-Z\beta))+B^2(4D\beta-\delta^2)}{4(4CD\beta-D\gamma^2-C\delta^2)} - A = \pi^* - A$$

■

The above proposition has a very important managerial implication. If a producer set X as above; then optimal values of all the decision variables are independent of both X and Y and have exactly the same form as in case 1. It should also be noted that X and Y are related. So a producer can first determine the independent variable Y (rate of change of s with respect to m) and can easily determine optimal X_{ms}^* . As we shall discuss later, it is not always necessary for the producer to consider case 2.

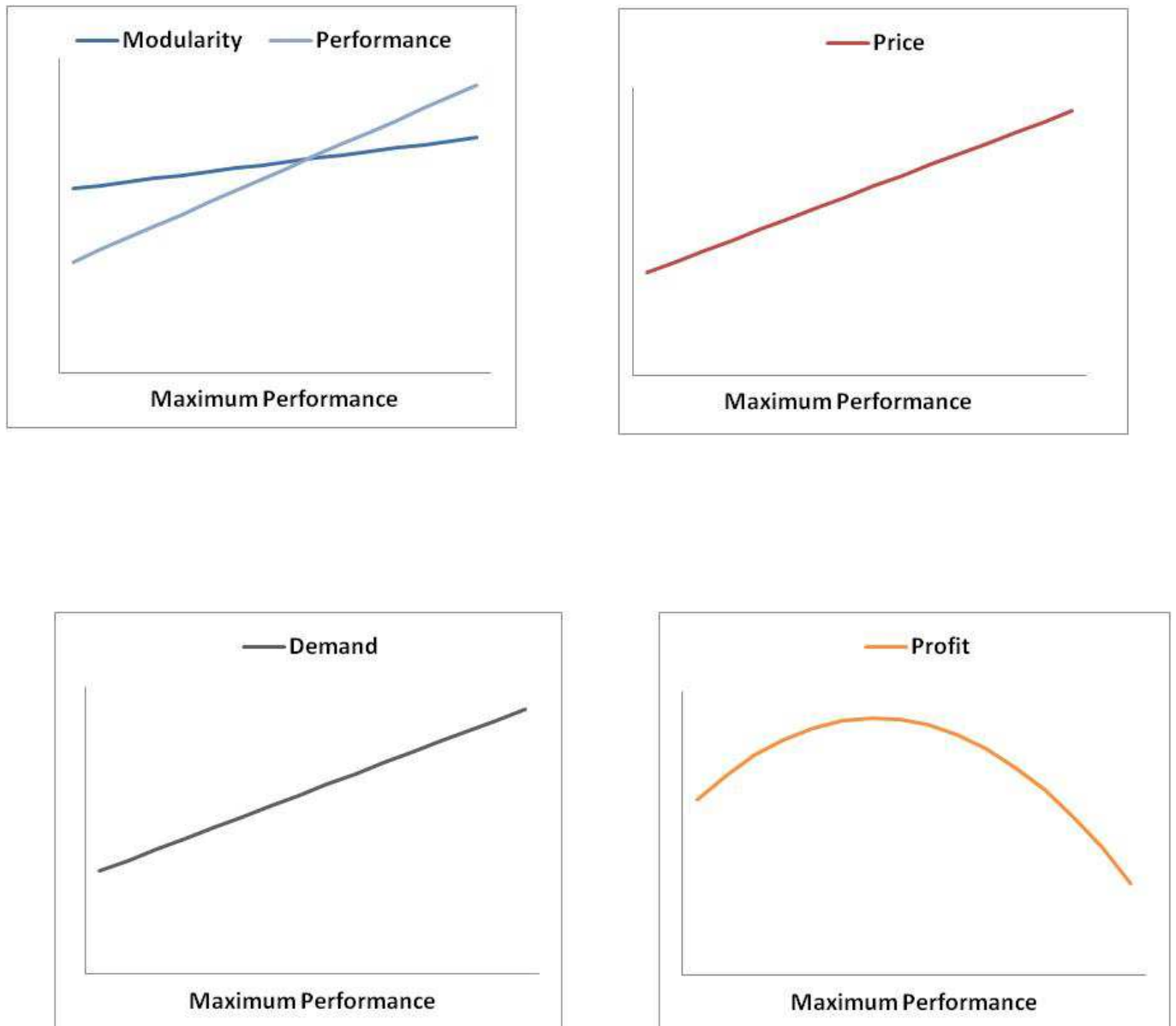


Figure 2.4: Graph showing how modularity (m), performance (s), price (p), demand and profit change with respect to change in maximum architectural performance.

The above graph also shows that unlike some other decision variables, increase in total possible architectural performance does not necessarily lead to increase in profit.

2.7 Discussions of Results and Implications

We discussed the results while we presented them. However, some of the results we obtained have significant managerial implications.

In our modeling we assumed two different cases. In case 1, we assumed that there is no relationship between modularity and performance. In case 2, we assumed an inverse relationship between modularity and performance. One of the most significant findings in this essay is that we found a condition where there is no difference in the values of parameters between the above two cases. As the case 2 is more realistic, it has an immense managerial implication. For maximizing profit, if the producer is able to vary both X and Y freely, then the optimal modularity and performance levels are identical to as in case 1 (architectural performance and modularity are unconstrained) and they are independent of X and Y. So it is not necessary for a provider to include X and Y in the modeling, unless it is not feasible to build an application with the optimal performance and modularity levels suggested in case 1; only then case 2 (performance and modularity are related inversely) becomes relevant. In that case, it is possible to estimate X and Y corresponding to feasible maximum modularization and performance levels in the IS application architecture. Although it will be possible to fix X and Y as suggested in proposition 7, however the producer could try to come as close to it as possible. Apart from this valuable insight, we also described how

optimal values change with respect to change in different values in parameter.

The importance of cloud computing in the IS area is now well established. This essay is an attempt towards understanding the phenomena from an economics perspective without making any simplistic assumption. We built a robust economic model of cloud computing and this model could be extended in many different ways.

First, in our model we included two very important non-functional attributes which could be easily operationalized for future research. Second, we introduced the concept of two dimensional performance – architectural and operational. Third, we recognized the service aspect of a SaaS application by including marginal cost in our model. Fourth, we uncovered the important roles modularity and architectural performance play in a SaaS application architecture design and their relationship with the profit of a SaaS provider. It has been suggested that there is an inverse relationship between modularity and performance (Clark, 1982). We analyzed two cases; we assumed in the first case that modularity and architectural performance are independent of each other and in the second case modularity and architectural performance are inversely related. We compared the results from both the cases and we observed that in most cases for determining optimal values of the decision

variables it is not necessary to consider the relationship between modularity and architectural performance.

2.8 Limitations

Although our models are more realistic as we do not ignore the service aspect of cloud computing, as in every research we have several limitations. First, we have not accounted completely the pricing structure of a service. Instead, we amortized the pricing over the lifetime of the product. Second, we only focused on the architectural performance that is relevant only to the architecture of our application. We excluded the effect of operational performance during the delivery or functional phase of the service. Third, we also did not account for competition in the market place. Fourth, we considered a linear demand function and it may not be very realistic.

2.9 Future Directions

This research could be extended in many different ways. First, some of the limitations discussed in previous section could be addressed in the extended model. Second, it will be interesting to investigate when there is more than one producer in the marketplace how does it affect profit maximization. Third, another way to extend this research will be to consider the effect of

complimentary services. Fourth, operational performance could be included in the model.

Chapter 3: Essay Two - Demand Planning for Cloud Computing: Effect of Random Variation in Demand

3.1 Introduction

In the previous chapters we uncovered many differences between a traditional IS application and a cloud enabled IS applications. We noted that traditional IS applications were customized products that were developed to solve a specific business problem for an organization. On the other hand, cloud computing enabled IS applications are a combination of (not customized) product and service. One of the main differences between a traditional IS application and a cloud enabled IS applications is how the users use it. Traditionally, the responsibilities of IS application developers are to uncover requirements for the proposed IS application and based on those requirements to develop the IS application; users are responsible for building the IT infrastructure and then to install and maintain the IS applications. In the context of cloud computing, the developers are also responsible for building the IT infrastructure and for installing and maintaining the IS application. In most cases it is consumers' responsibility to arrange for hardware and other infrastructure. Hence in cloud computing, it is important to recognize the service aspect of IS applications (Demirkan H. , 2008; Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010; Demirkan, Kauffman, Vayghan, Fill, Karagiannis, & Maglio, 2008).

However there is a paucity of research in service science in general and we are not aware of any specific research in the IS area that focuses on the service perspective of an IS application. Before the advent of cloud computing it was not necessary to consider demand planning for software vendors, as installation of infrastructure and day to day maintenance of IS applications were the responsibilities of consumers. Hence, demand planning was not important during the development of an IS application and researchers did not pay much attention to this topic in the context of IS applications. This work is an attempt to fill the void in this area. Here, we study how a SaaS application provider will determine the capacity of the IT infrastructure that needs to be planned for providing services at acceptable level to the users.

3.2 Motivation and Research Questions

During our previous discussions, we noted that one of the fundamental differences between a traditional IS application and a Cloud computing solution such as a SaaS application is that the latter is a combination of product and service. In the literature, there is some confusion in how product is defined. The words products and goods are often used interchangeably; however in the marketing literature product is defined as a combination of goods and services (Scheuing, 1989). We consider products

and goods as same. In order to plan for the capacity of the IT infrastructure for a SaaS application we look into the literature for both products and services and we first identify the key differences between a product and a service that is relevant for our specific problem.

Goods could be defined as “tangible economic products that are capable of being seen touched and may or may not be tasted, heard or smelled” (Rathmell, 1966, p. 32). Service is defined as a relationship between a producer and a consumer that creates and captures value and where the consumer participates actively (Gadrey, 2000; IBM; Fitzsimmons & Fitzsimmons, 2004). In other words, in the case of services, the consumers could be considered as co-producers. Another important characteristic of service is simultaneity of production and consumption or in other words, production and consumption of services occur at the same time. For products, consumers can wait to receive products and products could be produced and stored in an inventory, however for a service storage is not an option (Menor, Tatikonda, & Sampson, 2002; Rust & Chung, 2006). Hence, it is necessary for SaaS application providers to build IT infrastructure of appropriate capacity so that they are able to provide the service to the consumers. This brings up an additional challenge for SaaS application providers; they have to plan and make arrangements for infrastructure during development of SaaS applications in addition to deciding on optimal pricing.

Although capacity planning could involve many issues (Menasce & Ngo, 2009); we posit that demand planning will be one of those. Although demand could be predicted on the basis of price and other attributes, it is likely that there could be random variation in demand. So a vendor may face two different scenarios. In the first case, the actual demand is higher than the anticipated demand. In that case, a vendor will not be able to provide service to all the potential customers and will lose potential revenue. It is also possible in that situation potential customers will be forced to obtain service from a competitor and hence the service provider will not only lose the potential additional revenue but also they will lose any chance of future revenue. So we can conclude that it is possible that a vendor's loss could be even more than just the potential revenue. In the second case, if the demand is lower than the anticipated demand, the service provider will not be able to use all its capacity and will unnecessarily incur cost for over capacity.

Many SaaS providers use cloud infrastructure services for their needs instead of making capital expenditure and purchasing and managing their own hardware. As an example, social networking company FourSquare uses Amazon's Elastic computing services. Even in that case accurate planning of capacity is important. Amazon provides significant saving to its customers

who could predict their need accurately and purchase reserved instances of Amazon's Elastic computing services.

We introduce a term planned capacity. Planned capacity is defined as the capacity a producer should plan for and in most cases that will be different from optimal demand. We uncover the reason behind it. As we discussed earlier, we identified two dimensions of the attribute performance – architectural and operational; operational performance is the performance as observed by a user of a SaaS application (Transaction Processing Performance Council (TPC), 2012). We take an innovative two step approach for accurate prediction of planned capacity. First, we calculate optimal price and optimal demand using profit maximization model for producers. Second, we model a stochastic profit maximization problem where demand follows a certain probability distribution function; we assume that the mean of such probability distribution is equal to the optimal demand that we obtained from the previous step. We assume that there is no change in price because of random variation in demand, and the producers will charge the optimal price that was obtained in previous step. We then investigate how the planned capacity changes under various circumstances. We specifically focus on the following research questions.

1. How planned capacity (to be provisioned by a cloud computing application provider) would change from random variation in demand, when the random variation is small with respect to optimal demand?
2. How planned capacity depends on different strategies used by a SaaS application provider?
3. How planned capacity is related to operational performance of the service?

The rest of the essay is organized in the following way. In the next section we present a review of relevant literature. We then formulate our model for prediction of planned demand, followed by presentation of results of theoretical analysis. The last section discusses the results and presents possible future work in the area.

3.3 Literature Review

In the first essay we discussed in detail the importance of service component in a cloud computing application. The importance of demand planning in a cloud computing application arises from its service perspective. It is possible to delay a product shipment if there are too many orders, if it is acceptable to the consumers. However, for a service, a delay is not feasible and lower capacity could bring up many negative consequences.

Academic research in the general area of service management started a long time ago (Rathmell, 1966). However, because of numerous technological innovations service management has gone through a paradigm shift. Service always used to involve human interaction; however inventions such as self-service kiosk in the airports have changed that (Bitner & Brown, 2006). Spohrer et al discussed that the service science area lacks standards; they also discussed some of the important ideas that could lead to a theory of service systems (Spohrer, Maglio, Bailey, & Gruhl, 2007). Parasuraman et al introduced a scale for assessing quality of electronic service (Parasuraman, Zeithaml, & Malhotra, 2005). Rust and Chung discussed the importance of development of service models which could help efficient management of services (Rust & Chung, 2006). Maglio and Spohrer emphasized that service-dominant logic should be the philosophical basis of service science (Maglio & Spohrer, 2008).

The research in development of service on the other hand is rare; there are only a few works that discusses development of service in the context of IS. Never the less, the issue of service orientation has become very important in the IS area because of Service Oriented Architecture (MAS Research Roadmap Project, 2005). Cowell opined that although most of the western economy is service based, New Service Development (NSD) has been neglected in the literature (Cowell, 1988). According to Menor et al "Until

recently, the generally accepted principle behind NSD was that “new services happen” rather than occurring through formal development processes” (Menor, Tatikonda, & Sampson, 2002, p. 136). Magnusson et al observed that user involvement during service innovation is beneficial (Magnusson, Matthing, & Kristensson, 2003). Bolton et al investigated maintenance of business-to-business service relationship (Bolton, Smith, & Wagner, 2003). Hull investigated whether concurrent product development method is also applicable to services development. He found that service development could also benefit from the concurrent product development method model (Hull, 2004). Heim and Sinha presented a taxonomic analysis of Electronic Food Retailers (Heim & Sinha, 2002; Heim & Sinha, 2005).

Capacity planning has also been investigated in telecommunications area. Advent of IS based planning system made it possible to achieve productivity improvement from efficient capacity panning. Smunt investigated efficacy of learning curve analysis for capacity planning (Smunt, 1996). Laguna developed an Excel based decision support system for telecommunication providers to help them plan for expansion (Laguna, 1998). Papazoglou and den Heuvel introduced a web services management framework that included capacity planning (Papazoglou & den Heuvel, 2005). Ueno and Tatsubori emphasized the need for capacity planning for IS applications built using SOA architecture during early stages of system development lifecycle; they

investigated capacity planning of an Enterprise Service Bus in a web services based IS application (Ueno & Tsubori, 2009). Zhang et al investigated capacity issues while modeling price competition between two web services based application providers (Zhang, Tan, & Dey, 2009). Li and Lee investigated capacity planning in the context of pricing of peer-produced services for online communities (Li & Lee, 2010).

3.4 Model Formulation

As mentioned in the previous section, we model the problem in two steps. In the first step, we need to determine the demand level where the producer should set their initial demand forecast. In this step, we assume that there is no uncertainty in the demand. We first model our demand function. We assume a monopolist vendor and we consider a fixed period during which the IS application is being offered. Consistent with the literature, we assume that operational performance has positive effects on demand.

We assume a linear demand function and we include the sensitivity of demand related to price and operational performance

$$d = \alpha - \beta p + \delta o \quad (1)$$

where p is price of the SaaS application amortized over its lifetime, o is the operational performance level of the SaaS application.

α is primary demand due to functional attributes of the SaaS application and other non-functional attributes such as quality (except operational performance), brand image, performance, general economic fact that are outside the scope of this essay.

β represents price sensitivity of the demand, δ represents increase in demand from increase in operational performance. α , β , and γ are assumed to be greater than zero.

Our next step is to formulate the cost function. Our cost consists of three parts i.e. fixed cost, maintenance cost amortized over the lifetime of the service, and marginal cost per product also amortized over the lifetime of the product. We assume that fixed cost arising from increased operational performance to be a quadratic function of operational performance (o). This is in line with the standard practice in the IS literature; fixed costs incurred to improve quality of a product is a convex function of the slope of product improvement curve (Choudhary, 2007). We assume that cost (C_1) consists of fixed as well as the variable maintenance cost amortized over the lifetime of the service. Therefore C_1 can be expressed as:

$$C_1 = A + D o^2 \quad (2)$$

Where, A is the fixed cost arising from factors other than operational performance and it also includes general amortized maintenance cost over the lifetime of the product, D is the parameter related to operational

performance during initial setup of the service that cannot be changed very easily. As an example, in order to improve performance it is possible to increase capacity of the servers and the underlying hardware could be replaced. However, other infrastructure such as the room where the servers would be installed cannot be changed easily.

Unlike a traditional software vendor, a cloud computing application provider will also incur marginal cost for providing services. This marginal cost (C_2) will include both the cost for setting up infrastructure such as hardware, software as well as the amortized cost for providing the service. We further introduce a scaling factor ω ($\omega < 1$); and we assume that ωZ represents the amortized infrastructure cost necessary for setting up the service; $(1 - \omega) Z$ represents the variable cost, per unit of service offered. We assume that service is being set up for a capacity d and we also assume that the actual demand is also d . However we shall show later that depending on the variation in demand, the first part will remain unchanged whereas second part will change. Also the random demand could be less than planned capacity; however if it is larger than the planned capacity then the providers will not be able to meet the total demand. We assume that d is not greater than the planned capacity and we obtain the following equation.

$$C_2 = d \omega Z + d (1 - \omega) Z = d Z \quad (3)$$

where d is the number of instances that could be serviced by the vendor or in other words the demand subject to the constraint as discussed above, and Z is the marginal cost per application amortized over the lifetime of the product. Adding (2), and (3), our total cost function can be expressed as:

$$\theta = A + D o^2 + Z d \quad (4)$$

Profit for a producer (π) could be represented as:

$$\pi = d p - \theta$$

Using (1) and (5), the above profit can be rewritten as:

$$\pi = (\alpha - \beta p + \delta o)(p - Z) - (A + D o^2) \quad (5)$$

Boundary Condition:

We assume the boundary condition that *Primary demand of a product α is greater than product of marginal cost Z and price sensitivity β or $\alpha > \beta Z$*

In any IS application, it is fair to assume that functional attributes are much more important than non-functional attributes. It is unlikely that importance of non-functional attribute performance will be more than sum of functional attributes and other non-functional attributes (except performance). It is unlikely that a producer will produce a product based on only performance and without any regard to its functionality. We consider a case where performance sensitivity (δ) is close to zero or in other words, the customers

do not care about the product's performance. Following equation (1), we can rewrite the demand function as

$$d = \alpha - \beta p$$

In the above case, it is unlikely that a producer will produce product which does not have a projected positive demand. Hence, we assume that in this simplistic situation also demand should be positive. Hence, we obtain

$$\alpha > \beta p$$

As Z is the unit marginal cost, it is fair to assume that p must always be greater than Z , since otherwise it does not make sense for a producer to produce any product. As we are modeling the situation during design and production phase, we exclude the possibility of a fire sale. Hence it follows

$$\alpha > Z\beta \tag{6}$$

We shall assume the above boundary condition in our model.

3.5 Theoretical Results

In this section, we present our results. Our objective is to find optimal price (p), and architectural performance (ρ) that will maximize the profit function (5), subject to the boundary condition 6.

After making some simplification, we find the optimal values for our decision variables

$$p^* = \frac{2D(\alpha+Z\beta)-Z\delta^2}{4D\beta-\delta^2} = \frac{2D(\alpha-Z\beta)}{4D\beta-\delta^2} + Z \quad (7)$$

$$o^* = \frac{(\alpha-Z\beta)\delta}{4D\beta-\delta^2} \quad (8)$$

and that leads to optimum demand

$$d^* = \frac{2D\beta(\alpha-Z\beta)}{4D\beta-\delta^2} \quad (9)$$

and to optimum profit

$$\pi^* = \frac{D(\alpha-Z\beta)^2}{4D\beta-\delta^2} - A \quad (10)$$

We observe a relationship between optimal demand (d^*), price (p^*) and performance (s^*)

$$d^* = \frac{2D\beta o^*}{\delta}$$

$$p^* = \frac{d^*}{\beta} + Z = \frac{2D o^*}{\delta} + Z$$

We also observe a relationship between optimal demand (d^*) and profit (π^*)

$$\pi^* = \frac{d^* (\alpha - Z\beta)}{2\beta} - A$$

Lemma 1:

All the optimal decision variables as well as the optimal demand have positive values when $4D\beta > \delta^2$.

Proof:

The above condition is derived from the hessian matrix. The Hessian matrix is the second-order partial derivative of the profit function with respect to the variables p and s in the present case. The Hessian matrix H is shown under.

$$H = \begin{pmatrix} -2\beta & \delta \\ \delta & -2D \end{pmatrix}$$

A matrix is negative semi definite when its leading principal minors of different orders alternate in sign, starting with negative for the first leading principal minor of order 1 (Winston, 1993). Hence, we obtain the two following conditions:

$$\begin{aligned} -2\beta &< 0 \\ 4D\beta - \delta^2 &> 0 \end{aligned}$$

The first condition is trivial, since β is positive. To ensure that second condition is met, we assume

$$4D\beta > \delta^2 \tag{11}$$

Assumption of equations (11) along with the boundary condition equation (6) ensures that both numerators and denominators of equations (7), (8) and (9) are positive. ■

In a world where there is no random variation in demand, a provider of a SaaS application will plan for a capacity that is equal to the optimal demand as shown in equation (8). Such provider will charge the customers an optimal price as given in equation (6). However, in actual cases, demand will include a random component. Next, we examine how the demand uncertainty would impact producer's profit. We assume that random demand of the product is x , and it is distributed with a probability distribution function (pdf) $f(x)$ and cumulative distribution function (cdf) $F(x)$. We also assume a service provider has planned for a capacity Q instances of the service. If actual demand turns out to be less than the planned capacity Q , the service providers will not get appropriate returns on their investment. On the other hand, if the random demand x is greater than capacity Q , the service providers will miss out on additional profit as they will not be able to serve all the potential customers. We introduce a variable opportunity cost (u) per missed customer into our model. The opportunity cost measures the cost to the service providers when they miss out on making profit because they planned for a lower capacity. If the random demand is x and the planned capacity is Q and where $x > Q$, we assume that the total opportunity cost will be $(x-Q)u$. Different strategies could be used for modeling u . The opportunity cost (u) could be considered as the difference between price (p) and total marginal cost (Z).

So, we assume

$$u = (p-Z) \quad (12)$$

However, there could be several additional factors that could make above formulation of u inaccurate. Inability to provide service because of lower planned capacity could be detrimental to the interest of a service provider in many different ways. First, the service provider will not be able to earn additional profit and failure to earn additional profit could be formulated as cost. Second, the provider can incur loss of goodwill. As a prospective customer could not be served because of less capacity, it is possible that the customers may decide to go with another producer. In that case, the potential loss to a producer will be much more than $(p-Z)$. In this essay, we assume a very simple formulation for u as given in equation (12).

Proposition 1:

In a market where customer demand is such that it is subject to small random variation, the optimal capacity Q^ a producer should plan for, satisfies the following relationship and in most cases it is different from average demand.*

$$F(Q^*) = 1 - \frac{Z\omega(4D\beta - \delta^2)}{2D(\alpha + Z\beta + 2\beta u) - \delta^2(Z+u)} \quad (12)$$

Proof:

As we discussed above, we assume that random demand x is distributed with a pdf $f(x)$ and cdf $F(x)$. We assume that the average demand (expectation of demand) is same as optimal demand d^* as given in Equation (9).

$$E(x) = d^* = \frac{2D\beta(\alpha - Z\beta)}{4D\beta - \delta^2} \quad (13)$$

Third, we also assume that random variation in demand is much smaller compared with the actual demand. This assumption is very important and necessary so that we can still use the optimal values of the decision variables we derived, without considering random variation in demand. This assumption is mathematically represented as

$$V(x) \ll E(x) = d^* \quad (14)$$

Assuming random demand x is distributed with a pdf $f(x)$ and cdf $F(x)$, the expected demand $E(x)$ can be calculated as,

$$E(x) = \int_a^b x f(x) dx = \int_a^b x \left(\frac{dF(x)}{dx} \right) dx$$

Where a and b are the minimum and maximum values the demand variable x can take. We assume that theoretically the lowest and highest demand levels are 0 and infinite; the above expression can be rewritten as

$$E(x) = \int_0^{\infty} x f(x) dx = \int_0^{\infty} x \left(\frac{dF(x)}{dx} \right) dx \quad (15)$$

Next, we derive the function for random profit using the random demand x . We have to recognize some constraints that should be included in the model. Let us assume that a vendor has planned for a demand Q and that is different from d^* . First, at a maximum, vendors can only sell up to the capacity they have planned for. Second, if the random demand is less than the planned capacity, a vendor will still incur the cost of setting up the infrastructure for the planned demand Q . Hence, a vendor will always incur the cost $Z\omega Q$ no matter what the random demand x is; however, the other part of the marginal cost is only proportional to the random demand x . We also assume an opportunity cost u . As we discussed earlier, an opportunity cost arises when the demand is greater than the capacity and a vendor could not provide the service because of not having enough capacity. Hence, the random profit of a vendor for a maximum capacity Q could be formulated as

$$\pi(Q, x) = p^* \min(Q, x) - u \max(x - Q, 0) - Z\omega Q - Z(1 - \omega) \min(Q, x) - (A + D(s^*)^2) \quad (16)$$

We note that the first term gives the actual revenue and we have ensured that actual revenue never exceeds $p^* Q$. The second term describes the opportunity cost for lost revenue when $x > Q$. The third term describes the

marginal cost arising for setting up the infrastructure for a capacity Q . The fourth term describes the marginal cost that is only dependent on random demand x and we ensure that we cannot offer service greater than Q . The last term within bracket is the fixed cost of developing the service.

In order to find optimal value of Q , many different strategies could be taken. We consider the strategy of profit maximization by the producer. We assume that the producer will try to maximize average profit. We note that there are several other strategies that could be used, such as minimization of cost, minimization of loss of goodwill etc. Keeping Q constant, we can find the expectation of the profit (average profit) using equation (16) as

$$\begin{aligned}
 E[\pi(Q, x)] = & \\
 & p^* \int_0^Q x f(x) dx + p^* Q \int_Q^\infty f(x) dx - u \int_Q^\infty (x - Q) f(x) dx - Z\omega Q - \\
 & Z(1 - \omega) \int_0^Q x f(x) dx - Z(1 - \omega)Q \int_Q^\infty f(x) dx \\
 & - (A + D(\sigma^*)^2) \int_0^\infty f(x) dx \tag{17}
 \end{aligned}$$

$$\begin{aligned}
 = & p^* \int_0^Q x f(x) dx + p^* Q \int_Q^\infty f(x) dx - u \int_Q^\infty (x - Q) f(x) dx - Z\omega Q - \\
 & Z(1 - \omega) \int_0^Q x f(x) dx - Z(1 - \omega)Q \int_Q^\infty f(x) dx - (A + D(\sigma^*)^2) \\
 = & (p^* - Z(1 - \omega)) \int_0^Q x f(x) dx + (p^* + u - Z(1 - \omega)) Q \int_Q^\infty f(x) dx \\
 & - u \int_Q^\infty x f(x) dx - Z\omega Q - (A + D(\sigma^*)^2)
 \end{aligned}$$

$$\begin{aligned}
&= (p^* + u - Z(1 - \omega)) \int_0^Q x f(x) dx + (p^* + u - Z(1 - \omega)) Q \int_Q^\infty f(x) dx \\
&\quad - u \int_0^\infty x f(x) dx - Z\omega Q - (A + D(o^*)^2)
\end{aligned}$$

We recognize $\int_0^\infty x f(x) dx$ is the expectation of demand or average demand.

According to our assumption from equation 13, we have

$$d^* = \int_0^\infty x f(x) dx$$

Next, we simplify the expression of $E[\pi(Q, x)]$

We observe,

$$(p^* + u - Z(1 - \omega)) Q \int_Q^\infty f(x) dx = - (p^* + u - Z(1 - \omega)) Q (F(Q) - 1)$$

Using integration by parts, we can make the following simplification and we

note $f(x) = \left(\frac{dF(x)}{dx}\right)$

$$\begin{aligned}
&(p^* + u - Z(1 - \omega)) \int_0^Q x f(x) dx \\
&= (p^* + u - Z(1 - \omega)) \int_0^Q x \frac{dF(x)}{dx} dx \\
&= (p^* + u - Z(1 - \omega)) \left[x F(x) - \int F(x) dx \right]_0^Q \\
&= (p^* + u - Z(1 - \omega)) \left[Q F(Q) - \int_0^Q F(x) dx \right]
\end{aligned}$$

Therefore the expected profit can be rewritten as

$$E[\pi(Q, x)] = - (p^* + u - Z(1 - \omega)) \left(\int_0^Q F(x) dx \right) + (p^* + u - Z\omega - Z(1 - \omega)) Q - u d^* - (A + D(\sigma^*)^2) \quad (18)$$

In order to find optimal planned demand Q^* that will maximize the expected profit, we differentiate $E[\pi(Q, x)]$ with respect to Q and set it to zero

$$\frac{d}{dQ} (E[\pi(Q, x)]) = - (p^* + u - Z(1 - \omega)) F(Q) + (p^* + u - Z) = 0$$

From the above equation we derive a formula for optimal capacity Q^* that will maximize the average profit of a producer,

$$F(Q^*) = \frac{(p^* + u - Z)}{(p^* + u - Z(1 - \omega))} \quad (19)$$

$$F(Q^*) = 1 - \frac{Z\omega}{(p^* + u - Z(1 - \omega))}$$

Next, we test the consistency of the above equation. A cdf can only be in a range between 1 and 0. We carefully inspect equation 19. We know all the items are positive and ω is less than 1 and also positive. Hence the numerator is always less than the denominator. That ensures $F(Q^*)$ also varies between 0 and 1 as it should.

Next, substituting the value of p^* from equation (4) yields,

$$\begin{aligned}
 F(Q^*) &= 1 - \frac{Z\omega}{\left(\frac{2D(\alpha+Z\beta)-Z\delta^2}{4D\beta-\delta^2} + u - Z(1-\omega)\right)} \\
 &= 1 - \frac{Z\omega(4D\beta-\delta^2)}{2D(\alpha+Z\omega\beta+2\beta u)-\delta^2(Z\omega+u)} \quad (20)
 \end{aligned}$$

From equation we observe that there are only few cases where $F(Q^*) = .5$ or Q^* will be same as average demand. In most cases Q^* will be different from optimal (average) demand.

■

Although we could not obtain a closed solution for Q^* , we can still obtain some valuable insights from equation (20) by assuming how random demand will be distributed.

Lemma 2:

As the ratio of marginal infrastructure setup cost to marginal variable cost for providing service $\frac{\omega}{(1-\omega)}$ increases, the optimal capacity (Q^*) decreases if all other parameters remain same.

Proof:

As the value of ω lies between 1 and 0, we see as ω increases, numerator of $\frac{\omega}{(1-\omega)}$ increases and denominator of $\frac{\omega}{(1-\omega)}$ decreases and $\frac{\omega}{(1-\omega)}$ increases. So, we conclude $\frac{\omega}{(1-\omega)}$ increases, only when ω increases. From equation (19) we

observe, as ω increases the denominator increases and as a result of that $F(Q^*)$ decreases. Decrease of $F(Q^*)$ implies decrease of planned optimal capacity Q^* .

3.6 Numerical Results:

In order to get further insight into the equation 20 that shows a relationship between optimal capacity Q^* and other parameters we solve the equation numerically. We assume that the random demand x is normally distributed with a mean d^* and standard deviation $v d^*$, where v is a positive number much less than 1.

In order to calculate, optimal capacity it is necessary to calculate inverse of cumulative distribution function. We use the algorithm based on the algorithm suggested by Marsaglia (Marsaglia, 2004). The source code is in the appendix.

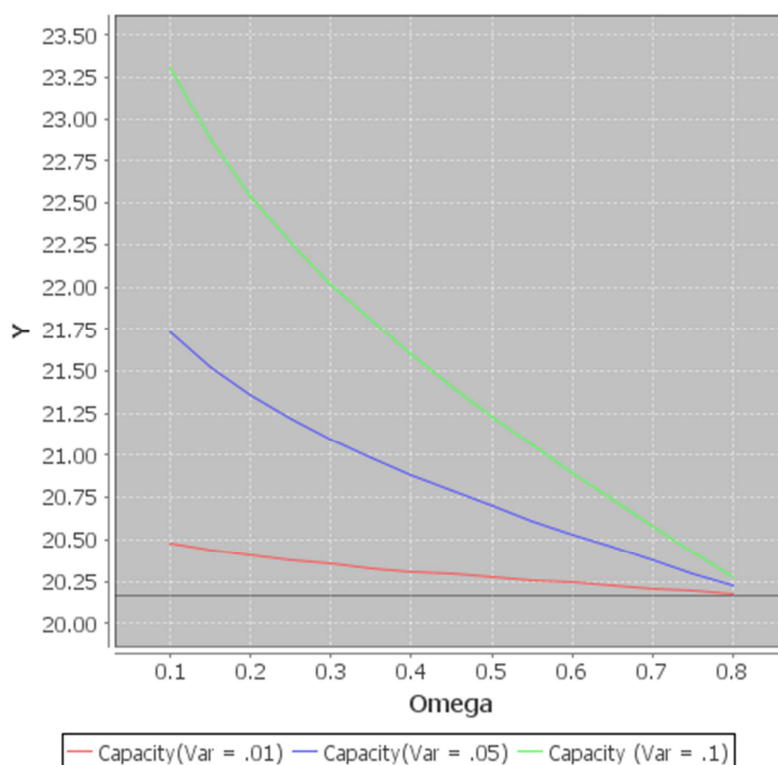


Figure 3.1: Graph showing the relationship between optimal capacity (Q) and ω

The relationship between ω and optimal capacity (Q) is shown above. We observe as ω increases optimal capacity decreases. However, in this case optimal capacity is always greater than optimal demand as shown by the red line. We keep the variance fixed at 1%, 5% and 10%. It shows for lower ω , optimal capacity changes with variance significantly. However, in all cases optimal capacity is greater than optimal demand (shown by the straight black line).

Next, we study the relationship between capacity and average profit and standard deviation of average profit. We set the variance in demand at 10%. We set ω at .3. As the capacity increases, the average profit first increases and then decreases. The average profit is maximum, when the capacity is equal to optimal capacity. However the standard deviation of profit keeps on increasing with increase of capacity.

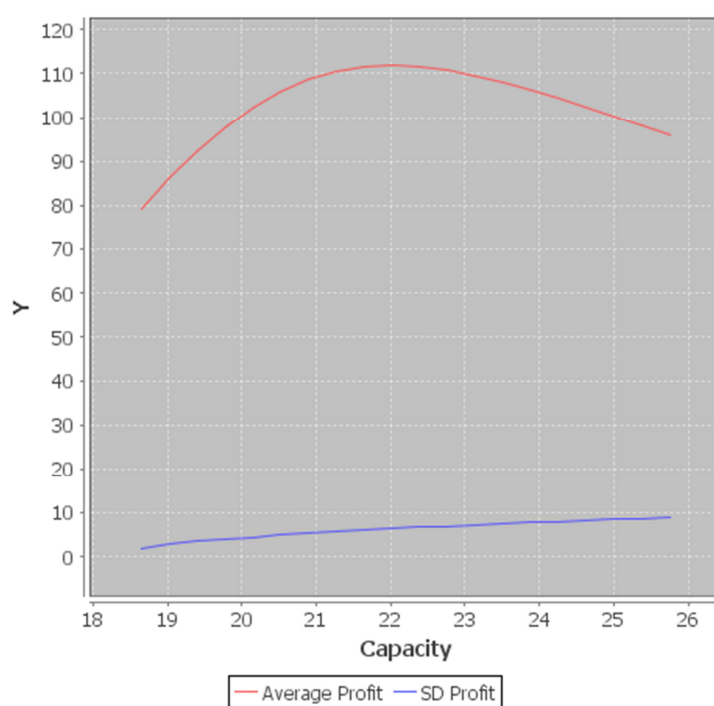


Figure 3.2: Graph showing the relationship between Average profit and Standard deviation of profit with respect to capacity

3.7 Discussions

We obtained important some important managerial implications from the above results. First, we uncovered that a SaaS application provider should always plan for a capacity which is greater than average demand in all cases. Second, we observed that as the ratio of cost of setting up infrastructure to variable cost for providing service increases the providers should plan for lower capacity. Third, we observed as a provider increases the planned capacity, the standard deviation of the average profit increases.

3.8 Limitations:

Some the limitations of this model are described here. First, in this model we have assumed that average demand is equal to the optimal demand. Although probably our assumption is correct, we have not offered any proof. Second, we have not fully developed the construct operational performance. In our model, we have not included any direct relationship between operational performance, random demand and capacity. However, we know that is not the case. For a fixed capacity decrease in random demand will lead to increase in operational performance. Third, we have used a single optimal value for operational performance. However, for most services the

providers offer different levels of performance and charge accordingly. Our model does not address that. Fourth, our model does not include that there could be more than one producers. We are sure that will have significant effect on capacity planning. Fifth, we have only showed numerical results where the random demand is normally distributed. Although normal distribution is probably the most appropriate one to use, we have not offered any support for that. Sixth, we formulated opportunity cost simplistically.

3.9 Conclusions and Contributions:

We recognized the service aspect of cloud computing. Unlike traditional products a service, cannot be stored in an inventory. Hence, the capacity planning is especially important in this context. We introduced a very innovative way of modeling services for the purpose of capacity planning. However, this is only a start. We discuss in the next section how this could be expanded. We envisage that using this approach many important managerial insights could be uncovered. We also showed a few important results. First, we clearly showed that in most cases optimal capacity is different from optimal demand. Second, for different parameters we showed the change of their effects on planned capacity analytically and numerically. Third, using numerical simulation we showed the consistency of our findings.

3.10 Future Directions

As we alluded to earlier this work could be extended in many different ways.

1. We took a very simplistic approach for coming up with random variation in demand. In order to make the model more realistic, it is necessary to introduce a concept of customers and each customer will have random variation in demand. That will give a more realistic and accurate picture.

2. We looked at operational performance as an abstract attribute. However, operational performance could be operationalized as transactions during a time fixed period. That will enable us to clearly model a relationship between performance and capacity. Hence, it should be investigated how increased capacity could lead to increase operational performance and increased profit. We completely neglected that in our model.

3. We also did not consider any tolerance in our demand. So our assumption is either a provider can provide a service or not. That is definitely not realistic.

Chapter 4: Essay Three - A Financial Risk Model for Cloud Computing

4.1 Introduction:

Most business decisions are made based on incomplete information and on assumptions which may not be accurate; as a result it is uncertain whether expected outcome will be achieved. In most cases, not achieving expected outcome could be considered a loss and hence it is fair to say that most decisions involve risk.

A cloud computing application provider is required to make many different decisions based on incomplete information during different phases of the product lifecycle such as price of the application, the infrastructure capacity, software architecture etc. Unlike other Information Systems (IS) applications, a cloud computing application is a mix of product and service and as a result of that vendors need to make more complex decisions and they have more responsibilities. As an example, it is essential for a vendor to make provisions for offering applications in such a way that the infrastructure is used efficiently and at the same time all the demands are met or in other words the vendors do not incur a stock out cost; in this essay we model the cost arising from stock out as opportunity cost. Hence, it follows that a cloud computing application provider has to take financial risk while making decision on capital investment of IT infrastructure. In addition to that, in the

literature it has been observed that IT investments in general are riskier than other capital investments (Dewan, Shi, & Gurbaxani, 2007). In this essay we develop a mathematical model that will enable cloud computing application providers to make decisions on capacity planning based on their risk tolerance.

Study of security and risk analysis in the IS area has a long history. The IS risk management takes an asset based approach and mainly focuses on risk emanating from data asset based on CIA Triangle and McCumber cube (National Security Telecommunications and Information Systems Security Committee, 1994) by identifying the vulnerabilities. Data Confidentiality model such as Bell-LaPadula security model was introduced in the early 1970s. Data Integrity models such as Biba Security models were introduced in late 1970 and Clark-Wilson Security model was introduced in the 1980s (Conklin, White, Williams, Davis, & Cothren, 2010). However, the risk in IS literature has mostly been studied by focusing on risk from security vulnerability or on risk associated during development of IS applications. These models only provide technical roadmaps for minimizing risks and these models cannot be used for estimating financial risks for cloud computing provider.

For quantifying risks in the IS area, cost benefit analysis is the most popular methodology (Whitman & Mattord, 2009). The cost benefit analysis gives a vague guidance whether or not to install a control; it does not offer any

customized guidance based on the risk tolerance of an application provider.

Hence, it is fair to conclude that no comprehensive model for IS risk analysis is available (Alter & Sherer, 2004; Yue, Çakanyıldırım, Ryu, & Liu, 2007). In the finance area, Modern Portfolio theory offers a clear roadmap for minimizing financial risk taking into account risk tolerance (Markowitz, 1959).

We posit that in order to solve the capacity planning problem, as a first step it is necessary to follow the risk management process and uncover the risks involved in the capacity planning process. We describe the risk management process in the following section. However, traditional risk management process needs to be modified in the cost benefit analysis phase by introducing risk tolerance concepts. We take the perspective of application developers and service providers. In this essay, we provide a framework that will help us study the following specific issues.

1. The effect of financial risk tolerance of SaaS application providers on capacity and price of the service.
2. The mediating effect of financial risk tolerance of cloud computing application providers on the relationship between different decision variables such as optimal capacity, optimal price etc.
3. The effect of modularity in software architecture on risk.

The rest of the essay is organized in the following way. In the next section, first we provide an introduction to the risk management area. Second, we do

a comprehensive review of relevant literature. Third, we develop theoretical financial risk model. Fourth, we numerically solve the financial risk model and present the results. Fifth, we discuss conclusions and finally we discuss possible extensions and contributions.

4.2 Introduction to Risk Management

Risk is defined as "potential harm that may arise from a future event, which may accrue either from incurring a cost ("downside risk") or by failing to attain some benefit ("upside risk")" (Wikipedia.org; Conklin, White, Williams, Davis, & Cothren, 2010; Whitman & Mattord, 2009).

Risk could be measured using the probability of occurrence of an undesirable event and possible loss incurred as a consequence of that (Katsikas, 2009).

However, measuring risk is difficult and inaccurate in general (Bojanc & Jerman-Blaz'ic', 2008). For measurement of risk qualitative, quantitative or a hybrid of both approaches could be taken. Specifically, quantitative evaluation of risk is more difficult (Bodin, Gordon, & Loeb, 2008). However quantitative assessment of risks is easier to interpret (Bashir & Christin, 2008).

Risk management is the complete decision making process which involves clearly identifying and possibly quantifying risks, identifying possible risk mitigation techniques and analyzing efficacy of those techniques (Peltier, 2005; Conklin, White, Williams, Davis, & Cothren, 2010). The main theme of IS risk management area has been security of data and vulnerabilities in an

IS application emanating from data insecurity. The most popular methodology is cost benefit analysis (CBA).

The IS risk management takes an asset based approach and mainly focuses on risk emanating from data asset based on CIA Triangle and McCumber cube (National Security Telecommunications and Information Systems Security Committee, 1994) by identifying the vulnerabilities. Next appropriate controls are identified that could eliminate or mitigate the risk. Finally, using cost-benefit analysis decisions are made whether or not to install the controls identified above and then monitoring is necessary to ensure that the process is working fine. The above phases are described below in detail (Bandyopadhyay, Mykytyn, & Mykytyn, 1999; Landoll, 2006; Mead, et al., 2009):

1. **Risk identification:** This is the first phase in the risk management. First, assets and important business processes are identified, classified and prioritized. Second possible threats to the assets are identified. Third, vulnerabilities in the assets are uncovered. Finally, possible impacts to the assets are identified when a threat is able to use the vulnerabilities in the asset and is able to successfully attack the asset.
2. **Risk assessment:** This phase involves quantifying risks using some metrics, prioritizing the risks and identifying the appropriate control methods. Both quantitative and qualitative approaches could be taken.

3. **Risk analysis:** This phase involves doing cost analysis of the possible controls using cost benefit analysis (CBA). The CBA involves estimating and quantifying the risks faced by an organization from business impacts of attacks, the probability of occurrence of attacks that could result from the vulnerabilities which were identified in the previous phase and how the proposed controls could reduce the possible loss and the cost of proposed control (Whitman & Mattord, 2009). Based on the results of cost benefit analysis (CBA) a final decision is made whether or not to implement the control.
4. **Risk mitigation:** This phase involves installation of appropriate controls for risk reduction as identified during CBA.
5. **Risk monitoring:** Finally this is the maintenance phase. Here appropriate data is collected and analyzed to ensure that the controls are meeting expectations and the initial assumptions made during CBA were accurate. If they are not then appropriate changes are made by identifying and analyzing the vulnerabilities.

It has also been noted that the above phases usually contain many atomic processes (Stoneburner, Goguen, & Feringa, 2002; Gregory, 2010). CERT has developed a framework for efficient risk assessment named Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) (Alberts & Dorofee, 2001). This framework was updated in 2007 and that was named OCTAVE Allegro (Caralli, Stevens, Young, & Wilson, 2007).

4.3 Literature Review

Risk management has been studied extensively in finance literature using different techniques (Focardi & Jonas, 1998) such as option pricing model, Modern Portfolio Theory (MPT) based on Markowitz's mean variance analysis model (Markowitz, 1959). Although in recent years many concerns have been raised about the validity of the mean variance analysis model (Taleb, Goldstein, & Spitznagel, 2009), it is still used extensively in finance for risk management.

In the Decision Science literature, risk is viewed as the probability distribution of outcomes both positive and negative; it has been suggested that for building systems which could help managers in risk management the concepts of Bayesian networks could be used (Miller, 2004). Markowitz's mean variance analysis model has also been used in the operation research area (Wu, Li, Wang, & Cheng, 2009). Yue et al studied the effect of system interdependence and layered protection strategies for IT security risk management (Yue, Çakanyıldırım, Ryu, & Liu, 2007).

Researchers in the IS area still use cost benefit analysis extensively during risk analysis (Arora, Hall, Pinto, Ramsey, & Telang, 2004), it has been observed that in many cases the traditional cost-benefit analysis based risk management is not adequate for making risk related decisions. Wang et al argued that use of traditional methods such as annual loss expectancy during cost benefit analysis could overlook important trends and they suggested use

of value-at-risk approach (Wang, Chaudhury, & Rao, 2008). Alternative methods such as use of option pricing models (Taudes, Feurstein, & Mild, 2000; Dewan, Shi, & Gurbaxani, 2007) or of Markowitz's mean variance analysis model (Wu & Ong, 2008), both adopted from the finance area have been found to be more effective.

4.4 Capacity Planning Framework:

We summarized the risk management process in section 4.2. As a first step towards capacity planning we need to follow those steps.

1. **Risk identification in capacity planning:** For this particular case, it is necessary to first uncover the risks which are involved in capacity planning. The two main risks are over capacity where a vendor plans for too much capacity and the infrastructure remains under-utilized and too little capacity where a vendor is unable to serve all the potential customers. In both cases, there could be many other factors in play such as dependence on another SaaS application, other security vulnerabilities etc. Those issues currently are out of the scope of this essay and we only focus on over or under capacity.
2. **Risk assessment in capacity planning:** This phase involves quantifying risks using some metrics. For over-capacity the additional cost incurred for excess capacity could be considered as potential loss. However, there may be many other factors which need to be considered. In some cases it may not be possible to plan for a very

specific amount. As an example, the servers come in specific sizes. It is not possible to purchase 1.5 servers. The same issues exist for under capacity. In this essay, we estimate under capacity as opportunity cost. How the opportunity cost is defined could depend on other risks also. It is also possible that some risks could be mitigated by planning for over capacity.

3. **Risk analysis in capacity planning:** This phase needs most modification and this is the main focus of our essay. We propose a quantitative analysis that includes risk tolerance.
4. **Risk mitigation in capacity planning:** This phase is pretty straight forward. This involves implementing the decisions made in previous phase. In our case, it involves installation of hardware application that measures actual usage levels etc.
5. **Risk monitoring:** The monitoring phase in capacity planning involves monitoring whether the decisions made regarding capacity need any further modification.

4.5 Modularity and Model Formulation

4.5.1 Modularity

Recently, there has been an immense shift in the architecture of data centers. In order to be competitive, organizations are introducing modularity into data center architecture. We recognize that, this phenomenon is very relevant to our case. As an example, IBM has developed a family of portable modular data center. It offers comprehensive data center assessment, design, build, and relocation services to meet the unique needs of any organization. It allows company to easily expand existing data-processing capability in remote or temporary environments. It also enables improved performance, higher density computing and greater cost-efficiency. It is vendor neutral and tries to minimize risk the customers will face because of technology obsolescence. So it is easy to see, by introducing modularity into data center architecture IBM is able to reduce risks for its customers.

We also note capacity planning involves risk. One of the most significant events in the area of cloud computing in recent history was the crash of Amazon's cloud. Social networking company FourSquare that depends on Amazon for providing service was totally unavailable for a period of time. In a later press release Amazon indicated among others a need to increase its infrastructure capacity to avoid future crashes.

The business examples above uncover a few relevant issues. First, it clearly demonstrates that decisions made about capacity planning could be risky. Second, modularity in architecture could lead to lesser risk. In chapter 3, we

investigated the relationship between optimal capacity and other decision variable under the condition of maximization of average profit. We observed an interesting trend in figure 3.2. We saw as a producer increases capacity, initially both average profit and standard deviation of profit increases till the capacity is equal to optimal capacity. After that as expected average profit decreases and standard deviation of profit increases. Although optimal capacity leads to the highest average profit that also leads to higher variance in profit and for some service provider higher variance in profit may not be acceptable as they may not like the additional risk.

Prior research has shown that introduction of modularity into product architecture leads to higher demand in products. Modularity has also enabled producers to offer mass customized product. Focusing on supply chain management area, Weng showed that modularity in product architecture reduced system cost by employing joint buffer stock for a group of products in a two-echelon distribution system with multiple retailers (Weng, 1999). Similarly, we posit that introducing modularity in architecture will enable a producer to use same module in more than one product and that would lead to reduction invariance in demand. We know if there are two variables X_1 and X_2 that are normally distributed with mean and standard deviation $X_1 (\mu_1, \sigma_1)$ and $X_2(\mu_2, \sigma_2)$ then the sum of them would be normally distributed as $X ((\mu_1 + \mu_2), \sqrt{\sigma_1^2 + \sigma_2^2})$. It can be shown that

$\frac{\sqrt{\sigma_1^2 + \sigma_2^2}}{\mu_1 + \mu_2}$ is less than maximum of $(\frac{\mu_1}{\sigma_1}, \frac{\mu_2}{\sigma_2})$ so it is fair to assume that if we can use same module for more than one product then the total variation of demand will be less than the maximum of original variation of demand in either cases. This theoretical background supports the conclusion found by (Weng, 1999).

4.5.2 Model Formulation

If a producer has more than one service offering and if a specific modular component could be used in more than one service then it might be easier to plan for capacity. Variation in demand of one product could be compensated by other product's variation in demand. A complete discussion of two products is beyond the scope of this essay. However, we shall recognize this specific property of modularity by including a parameter in the random demand function; more modular a service is, the standard deviation of its random demand decreases. Although, we understand this is a little bit simplistic assumption however it will give us important insights. Hence, we posit that modularity in software architecture of an IS application can play a special role in capacity planning. As a first step towards building risk model that includes modularity, we modify our demand function that we used in chapter 3.

Consistent with the literature and experiences from industry we assume that modularity has positive effects on demand. Joglekar and Rosenthal observed that use of modularity in software architecture improved outcomes of mainstream product which has added software components (Joglekar & Rosenthal, 2003). Modularity would also support mass customization strategy which allows producers to offer their products to a more diverse group of customers (Dewan, Jing, & Seidmann, 2003).

We assume a linear demand function and we include the sensitivity in demand from modularity

$$d = \alpha - \beta p + \gamma m + \delta o \quad (8)$$

where p is price of the SaaS application amortized over its lifetime,

m is the modularity level of the SaaS application

o is the operational performance level of the SaaS application

α is primary demand due to functional attributes of the SaaS application and other non-functional attributes such as quality (except modularity and operational performance), brand image, performance and general economic condition

β represents price sensitivity of the demand,

γ represents increase in demand from increase in modularity

δ represents increase in demand from increase in operational performance

α , β , γ and δ are assumed to be greater than zero.

Our cost consists of three parts i.e. fixed cost, maintenance cost amortized over the lifetime of the product, and marginal cost per product also amortized over the lifetime of the product. Prior research has shown that modularity in product architecture leads to higher product complexity (Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010). Hence we can infer production of modular software will require more production cost for vendors (i.e. higher upfront (fixed) cost). We assumed that fixed cost (C_1) arising from increased modularity to be a quadratic function of modularity (m). This is in line with the standard practice in IS literature; fixed costs incurred to improve quality of a product is a convex function of the slope of product improvement curve (Choudhary, 2007). Therefore C_1 can be expressed as:

$$C_1 = A_1 + C m^2 + D o^2 \quad (9)$$

where A_1 is the fixed cost arising from factors other than modularity and performance, C is the parameter related to modularity during design and development of the application, and D is the parameter related to operational performance.

Modularity in design also leads to better flexibility in changing products leading to agility (MacCormack, Verganti, & Iansiti, 2001). It was more

expensive to maintain a non-modular product compared to a modular product (Banker, Datar, Kemerer, & Zweig, 1993; Bardhan, Demirkan, Kannan, Kauffman, & Sougstad, 2010). We treat maintenance cost (C_2) as amortized over the lifetime of the product. Hence, C_2 can be expressed as:

$$C_2 = A_2 - B m \quad (10)$$

where A_2 is the general amortized maintenance cost over the lifetime of the cloud computing application and B is the parameter related to modularity showing the saving in maintenance cost arising from modular design also amortized over the lifetime of the cloud computing application.

Unlike a traditional software vendor, a cloud computing application provider will also incur marginal cost for providing services. This marginal cost (C_3) will include both the cost (C_{31}) for setting up infrastructure such as hardware, software as well as the cost (C_{32}) for providing the service. We recognize that for setting up infrastructure, a product with higher operational performance is more expensive. We recognize that by

$$C_{31} = d (G o + \omega Z)$$

$$C_{32} = d (1 - \omega) Z$$

where ω ($\omega < 1$) is a scaling factor; and we assume that ωZ represents the infrastructure cost necessary for setting up the service amortized as per

capacity and it excludes the additional cost that a provider will incur from higher operational performance; and $(1 - \omega) Z$ represents the variable cost, both are expressed per unit of services (demand) offered. $G m$ is the increase in setup cost from higher operational performance.

We assume that service is being set up for a capacity d and we also assume that the actual demand is d . However we shall show later that depending on the variation in demand, the first part will remain unchanged whereas second part will change. Also the demand could be less than planned demand; however it can never be more than the planned demand. Hence, C_3 can be expressed as:

$$C_3 = d G o + d \omega Z + d (1 - \omega) Z = d (Z + G o) \quad (11)$$

where d is the number of applications that is being planned, and Z is the marginal cost per application amortized over the lifetime of the product. However, we recognize that d is the projected demand and it will probably be different from actual demand. In that case, there will be a role played by ω . Adding (9), (10), and (11), our total cost function can be expressed as:

$$\theta = A - B m + C m^2 + D o^2 + (Z + G o) d \quad (12)$$

where $A = A_1 + A_2$

Profit for a producer (π) could be represented as:

$$\pi = d p - \theta$$

Using (1) and (5), the above profit can be rewritten as:

$$\pi = (\alpha - \beta p + \gamma m + \delta o) (p - (Z + G o)) - (A - B m + C m^2 + D o^2) \quad (13)$$

Our objective is to find optimal price (p^*), modularity (m^*), and operational performance (o^*) that will maximize the above profit function.

After making some simplification, we find the optimal values for our decision variables as:

$$p^* = \frac{2C(2D(\alpha + Z\beta) - (G\beta - \delta)(G\alpha - Z\delta)) + \gamma(-2DZ\gamma + B(2D + G(-G\beta + \delta)))}{2(4CD\beta - D\gamma^2 - C(\delta - G\beta)^2)}$$

$$= \frac{D(2C(\alpha - Z\beta) + B\gamma)}{4CD\beta - D\gamma^2 - C(\delta - G\beta)^2} + Z + G o^* \quad (14)$$

$$m^* = \frac{2D(\alpha - Z\beta)\gamma + B(4D\beta - (\delta - G\beta)^2)}{2(4CD\beta - D\gamma^2 - C(\delta - G\beta)^2)} \quad (15)$$

$$o^* = \frac{(2C(\alpha - Z\beta) + B\gamma)(\delta - G\beta)}{2(4CD\beta - D\gamma^2 - C(\delta - G\beta)^2)} \quad (16)$$

and that leads to optimum demand and profit

$$d^* = \frac{D\beta(2C(\alpha - Z\beta) + B\gamma)}{4CD\beta - D\gamma^2 - C(\delta - G\beta)^2} \quad (17)$$

$$\pi^* = \frac{4CD(\alpha - Z\beta)^2 + B(4D(\alpha - Z\beta)\gamma + B(4D\beta - (\delta - G\beta)^2))}{4(4CD\beta - D\gamma^2 - C(\delta - G\beta)^2)} - A \quad (18)$$

To ensure profit (π) has a local maximum the Hessian matrix needs to be negative semi definite.

The Hessian matrix H is shown under.

$$\begin{pmatrix} -2\beta & \gamma & G\beta + \delta \\ \gamma & -2C & -G\gamma \\ G\beta + \delta & -G\gamma & -2D - 2G\delta \end{pmatrix}$$

To ensure that profit has a local maximum the determinants of the Hessian matrix need to be negative semi definite. A matrix is negative semi definite when its leading principal minors of different orders alternate in sign, starting with negative for the first leading principal minor (Winston, 1993). Hence the principal minor of order 3 has to be negative. The only principal minor of order three is the determinant of the matrix itself which is $2(D\gamma^2 + C(-4D\beta + (-G\beta + \delta)^2))$, and that leads to the following condition.

$$4CD\beta > D\gamma^2 + C(\delta - G\beta)^2 \quad (19)$$

We note that producer would plan for demand assuming that they would charge their customers at price as given in equation (14). However, in many cases, demand is not constant. Therefore, in our second step, we examine how the demand uncertainty would impact producer's profit. Here we assume that random demand of the product is x and it is distributed with a probability

distribution function (pdf) $f(x)$ and cumulative distribution function $F(x)$ (cdf).

Hence the expected demand $E(x)$ can be calculated as,

$$E(x) = \int_a^b x f(x) dx = \int_a^b x \left(\frac{dF(x)}{dx} \right) dx$$

Where a and b are the minimum and maximum demand values x can take.

We assume that the lowest and highest demand levels are 0 and very large (infinite in mathematical terms); the above expression can be rewritten as

$$E(x) = \int_0^{\infty} x f(x) dx = \int_0^{\infty} x \left(\frac{dF(x)}{dx} \right) dx \quad (20)$$

We estimate that expected demand is the optimal demand d^* as given in equation (9).

$$E(x) = d^* \quad (21)$$

Next, we derive the function for random profit using the random demand x .

We have to recognize some constraints that should be included in the model.

First, at a maximum, vendors can only sell up to the capacity they have planned. Let us assume that a vendor has planned for a demand Q . On the other hand if the random demand is less than the planned demand Q , a vendor will still incur the cost of setting up the infrastructure for the demand Q . Hence, a vendor will always incur the cost $(Z\omega + G\sigma^*)Q$, no matter what the random demand x is; however, the other part of the marginal cost is only proportional to the random demand x .

We also assume an opportunity cost. An opportunity cost arises when the demand is greater than the capacity and a vendor could not provide the service because of not having enough capacity. We assume that the opportunity cost of revenue is u . In terms of exact formulation u could be considered as missed revenue $(p-Z)$. However, there could be several other factors that would determine u such as loss of goodwill. We consider opportunity cost as one of the costs.

Hence, the random profit of a vendor for a maximum capacity Q could be formulated as

$$\pi(Q, x) = p^* \min(Q, x) - u \max(x - Q, 0) - (Z\omega + G o^*)Q - Z(1 - \omega)\min(Q, x) - (A - B m^* + C (m^*)^2 + D (o^*)^2) \quad (22)$$

We note that the first term gives the actual revenue and we have ensured that actual revenue never exceeds $p^* Q$. The second term describes the opportunity cost for lost revenue when $x > Q$. The third term describes the marginal cost arising for setting up the infrastructure for a planned demand Q . The fourth term describes the marginal cost that is dependent on random demand x . The last term within bracket is the fixed cost for developing the service.

We recall that in chapter 3, we obtained an equation for optimal value of Q , by maximizing the average profit. Instead in this case we shall maximize the following expression suggested by Mean Variance Analysis model

$$\{E[\pi(Q, d^*, x)] - \epsilon \sqrt{V[\pi(Q, d^*, x)]}\} \quad (23)$$

The variance of the profit can be calculated using the following formula

$$V[\pi(Q, d^*, x)] = (E[\pi(Q, d^*, x)^2]) - (E[\pi(Q, d^*, x)])^2 \quad (24)$$

In the above case, we maximize average profit subject to a constraint that variation of average profit is less. How much importance we give to the variation in average profit depends on the factor ϵ .

We can see if $\epsilon = 0$, then it simplifies to just maximization of average profit.

However, we need a starting value for optimal capacity in order to make the numerical simulation process more efficient. Hence, we first find the optimal capacity under the simplified condition when $\epsilon = 0$. In that case, we need to first calculate average profit from equation 22 and then differentiate that with respect to capacity Q and set it to zero or in other words we maximize the average profit.

Keeping Q constant, we find the expectation of the profit (average profit) using equation (13) as

$$E[\pi(Q, d^*, x)] =$$

$$\begin{aligned} & p^* \int_0^Q x f(x) dx + p^* Q \int_0^\infty f(x) dx - u \int_0^\infty (x - Q) f(x) dx - (Z \omega + \\ & G o^*) Q \int_0^\infty f(x) dx - Z(1 - \omega) \int_0^\infty x f(x) dx - (A - B m^* + C (m^*)^2 + \\ & D (o^*)^2) \int_0^\infty f(x) dx \end{aligned} \quad (25)$$

$$\begin{aligned} & = p^* \int_0^Q x f(x) dx + p^* Q \int_0^\infty f(x) dx - u \int_0^\infty (x - Q) f(x) dx - (Z \omega + G o^*) Q \\ & \quad - Z(1 - \omega) \int_0^Q x f(x) dx - Z(1 - \omega) Q \int_0^\infty f(x) dx \\ & \quad - (A - B m^* + C (m^*)^2 + D (o^*)^2) \end{aligned}$$

$$= p^* \int_0^Q x f(x) dx + (p^* + u -$$

$$Z(1 - \omega)) Q \int_0^\infty f(x) dx - u \int_0^\infty x f(x) dx - (Z \omega + G o^*) Q - Z(1 -$$

$$\omega) \int_0^Q x f(x) dx - (A - B m^* + C (m^*)^2 + D (o^*)^2)$$

$$= p^* \int_0^Q x f(x) dx + (p^* + u -$$

$$Z(1 - \omega)) Q \int_0^\infty f(x) dx - u \int_0^\infty x f(x) dx + u \int_0^Q x f(x) dx - (Z \omega + G o^*) Q -$$

$$Z(1 - \omega) \int_0^Q x f(x) dx - (A - B m^* + C (m^*)^2 + D (o^*)^2)$$

$$= (p^* + u -$$

$$Z(1 - \omega)) \int_0^Q x f(x) dx + (p^* + u -$$

$$Z(1 - \omega)) Q \int_0^\infty f(x) dx - u \int_0^\infty x f(x) dx - (Z \omega + G o^*) Q - (A - B m^* +$$

$$C (m^*)^2 + D (o^*)^2)$$

$$=(p^* + u - Z(1 - \omega)) \left(\int_0^Q x f(x) dx + Q \int_Q^\infty f(x) dx \right) - u \int_0^\infty x f(x) dx - \\ (Z\omega + G o^*)Q - (A - B m^* + C (m^*)^2 + D (o^*)^2)$$

We observe,

$$Q \int_Q^\infty f(x) dx = Q (1 - F(Q))$$

$$E[\pi(Q, d^*, x)]$$

$$=(p^* + u - Z(1 - \omega)) \left(\int_0^Q x f(x) dx + Q (1 - F(Q)) \right) - u \int_0^\infty x f(x) dx - \\ (Z\omega + G o^*)Q - (A - B m^* + C (m^*)^2 + D (o^*)^2)$$

We recognize $\int_0^\infty x f(x) dx$ is the expectation of demand or average demand.

According to our assumption from equation 12, we have

$$d^* = \int_0^\infty x f(x) dx$$

Next, we simplify the expression of $E[\pi(Q, x)]$

Using integration by parts, we can make the following simplification and we

$$\text{note } f(x) = \left(\frac{dF(x)}{dx} \right)$$

$$(p^* + u - Z(1 - \omega)) \int_0^Q x f(x) dx = (p^* + u - Z(1 - \omega)) \int_0^Q x \left(\frac{dF(x)}{dx} \right) dx$$

$$= (p^* + u - Z(1 - \omega)) \left[x F(x) - \int F(x) dx \right]_0^Q$$

$$= (p^* + u - Z(1 - \omega)) (Q F(Q) - \int_0^Q F(x) dx)$$

Therefore the expected (average) profit can be rewritten as

$$E[\pi(Q, d^*, x)] = - (p^* + u - Z(1 - \omega)) (\int_0^Q F(x) dx) + (p^* + u - (Z(1 - \omega) + Z\omega + G o^*)) Q - u d^* - (A - B m^* + C (m^*)^2 + D (o^*)^2) \quad (26)$$

In order to find optimal planned demand Q^* that will maximize the expected profit, we differentiate $E[\pi(Q, x)]$ with respect to Q and set it to zero

$$\frac{d}{dQ} (E[\pi(Q, d, x)]) = - (p^* + u - Z(1 - \omega)) F(Q) + (p^* + u - (Z + G o^*)) = 0$$

From the above equation we derive a formula for optimal capacity Q^* which will maximize the average profit of a producer,

$$F(Q^*) = \frac{(p^* + u - (Z + G o^*))}{(p^* + u - Z(1 - \omega))} \quad (27)$$

As we discussed earlier our goal is to include risk in our model; we do so by maximizing the expression given in equation 23. It is difficult to solve the equation analytically; we use numerical simulation. We assume that the demand is distributed normally with a mean as given by optimal demand as in equation 17. We use the optimal capacity calculated using equation 27 as a starting value and we vary the planned capacity around that and we identify the planned capacity when the expression given in equation 23 is maximum. We also note that how much importance we give to the variance is determined by varying ϵ .

As we indicated in previous section, one of our main contributions is to introduce risk tolerance in the risk assessment phase. We use Markowitz's mean variance analysis model for the purpose. Modern Portfolio theory is based on Markowitz's mean variance analysis model; it is used for choosing a diversified portfolio, and in the model the risk is estimated by standard deviation of return. A specific concern that has been brought up regarding the accuracy of the model's assumption that the financial return is normally distributed; it has been argued that the assumption is invalid specifically during black swan events (Taleb, Goldstein, & Spitznagel, 2009). However, in our case assumption of normal distribution is realistic as there is evidence that demand does follow normal distribution.

We develop a computer program for performing numerical simulation using Java programming language in Eclipse IDE. We assume that demand is normally distributed with a mean optimal demand that we obtained earlier analytically. We approximate variance of the distribution as a percentage of the mean. Next, based on all the parameters we obtain the optimal capacity using equation 27. Equation 27 involves calculation of inverse cumulative distribution function. We use the algorithm suggested by Marsaglia (Marsaglia, 2004).

In the simulation, we define a range for the capacity using the optimal capacity. For each capacity, we perform 100,000 simulations with a random

demand that is normally distributed as described above. For each trial we calculate the profit using equation 22. Next, we calculate both mean and standard deviation of the profit. Finally for each capacity, we obtain the expression $\{E[\pi(Q, d^*, x)] - \epsilon \sqrt{V[\pi(Q, d^*, x)]}\}$ and we have named it Av Profit2 in the graphs. We find out the capacity when the above expression is maximum. We also vary ϵ and that represents risk tolerance of the providers. Greater ϵ implies that the providers are more risk averse.

4.6 Results

In this section, we first present the results of numerical simulation in a graphical format.

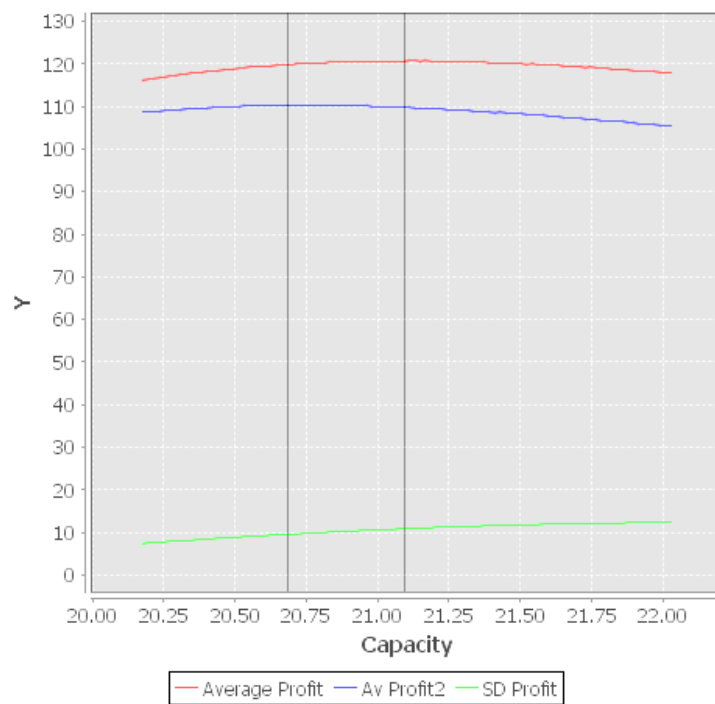
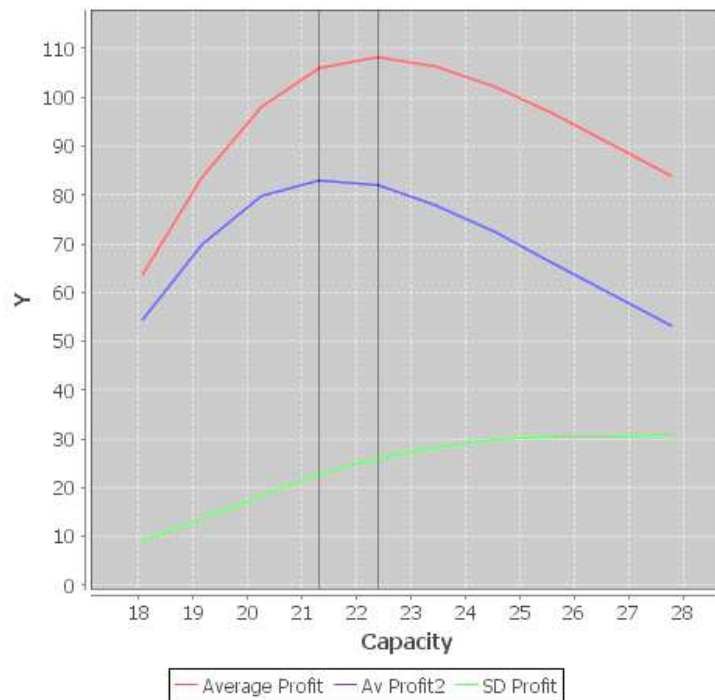


Figure 4.1: Graph showing the effect of change in planned capacity on Average profit for risk neutral (red line) and risk averse (blue line) providers and Standard deviation of profit

In both the above graphs we show the planned optimal capacity when average profit (indicated by red line) or the average profit factor that takes into account also the risk factor (indicated by blue line). Finally, the green line depicts the standard deviation of actual profit. The main difference between the two graphs is the numerical assumptions of the different constants within the allowed values. We can make the following general conclusions from the above graph.

1. First we note the relationship between planned capacity, average profit and standard deviation of profit. As we increase the planned capacity the average profit first increase and it has a maxima. Then the average profit decreases. However, the standard deviation of the profit consistently increases. It implies that our results are consistent with our expectations. There is an optimal capacity that maximizes the average profit. It is consistent with what we find out through numerical simulation. However, as planned capacity increases the variation in random profit increases consistently.
2. Next, we note that when we introduce a risk aversion factor in the profit the expression depicting average profit with risk averseness factor has a maxima corresponding to the planned optimal capacity that is less than the planned optimal capacity when risk is not considered. So a risk averse provider with generally plan for lesser planned capacity corresponding to the providers who are risk takers.

Next, we present how optimal planned capacity changes with ε that signifies risk averseness of a cloud computing provider and ω the ratio of fixed infrastructure cost variable infrastructure cost. Increase of ε signifies a more risk averse provider. Increase of ω implies that providers incur more cost while setting up the infrastructure compared with the variable cost for maintaining the infrastructure.

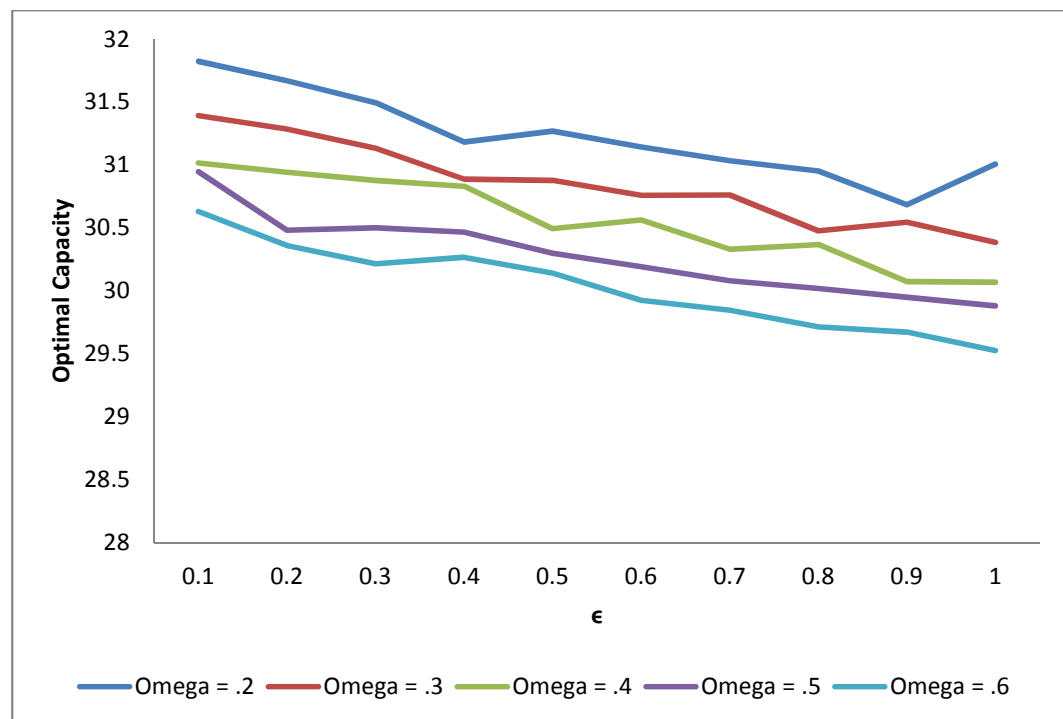


Figure 4.2: Graph showing the relationship between optimal capacity and risk averseness factor ε for different values of ω

We can make the following general conclusions from the above graph.

1. Increase in ϵ (increase in risk averseness) leads to decrease in optimal planned capacity. As a providers become more risk averse they plan for a lower infrastructure capacity.
2. Increase in ω (ratio of fixed infrastructure cost and variable marginal cost) leads to increase in optimal planned capacity.

4.7 Limitations

We have identified a few of the limitations. First, in this model we have not included any direct relationship between operational performance, random demand and capacity. We have also assumed that there is no relationship between random demand and operational performance. However, we know that is not the case. For a fixed capacity, decrease in random demand will lead to increase in operational performance in many cases depending on the software architecture. Third, we have used a single optimal value for operational performance. However, for most services the providers offer different levels of performance and charge accordingly. Our model does not address that. Fourth, our model does not include that there could be more than one producers. We are sure that will have significant effect on capacity planning. Fifth, we have only showed numerical results where the random demand is normally distributed. Although normal distribution is probably the most appropriate one to use, we have not offered any support for that. Sixth, we formulated opportunity cost simplistically.

4.8 Future research

Our research is a preliminary effort where we included the concept of financial risk in the IS area. First, the model needs to be expanded to include a relationship between operational performance, planned capacity and actual demand. It is intuitive to figure out that for a fixed planned capacity as actual demand increases the operational performance decreases. Hence the parameter operational performance needs to be researched more so that we get a more realistic understanding of its impact on other parameters. Second, we assumed that if the actual demand is greater than planned capacity then the provider will not be able to service additional customers. This assumption is unrealistic. There cannot be a sharp cutoff; instead a provider may be able to service the customers however the operational performance will be lower. Third, our random demand needs to be formulated in a better way. We assumed no variation in demand from a user; this assumption is simplistic. The model could be improved by taking into account random variation in demand from a particular user. This will have a considerable impact on the model as it will be unlikely that all the users will have a same demand pattern. This could be modeled pretty easily by representing demands as number of transactions during a particular amount of time. The planned capacity could be operationalized as the number of transactions during a fixed amount of time. This problem could be solved using numerical simulation where each user could be represented in separate threads. Fourth,

this research could be extended to a two product problem where each product will share a few modules. In that way effect of modularity on a product could be understood in a better way.

Chapter 5: Contributions

This dissertation is an effort to understand the phenomenon of cloud computing and the challenges and opportunities that it present to the researchers in the IS area. We believe we obtained many interesting results that will spur more research in the area. Also, practitioners will be able use these models after proper calibration for making decisions regarding pricing, capacity planning etc. Specific contributions have been identified below.

5.1 Research Contributions

We believe that the most important contribution of this dissertation is uncovering the changes that are happening from the advent of cloud computing.

We identified that as a result of the industrialization of IT, IS applications are changing from customized product (traditional IS applications) to a combination of product and service (cloud computing applications). Hence, it is necessary to include both marginal cost and maintenance cost in a model of cloud computing application; traditional IS applications were considered developmentally intensive products, and marginal costs were not included

there (Krishnan & Zhu, 2006). In all three essays, we included both marginal and maintenance costs.

We identified another important aspect of cloud computing – capacity planning. As we discussed earlier, this was never an issue in the context of traditional IS application for the application providers. Because of the service aspect, cloud computing providers have to estimate the possible demand and then install the necessary infrastructure for it. Our second and third essays addressed this problem and we developed an analytical model for calculating optimal capacity, using a two-step innovative analytical technique.

We observed that a cloud computing application provider needs to make many different decisions based on incomplete information; making decisions which involve taking risks. The third essay addresses this. We developed a model that would help cloud computing application providers make decisions on planned capacity based on their personal financial risk tolerance. We used Markowitz's mean variance analysis model (Markowitz, 1959) for this purpose. The mean variance analysis model has been used widely in the finance area for a different purpose. Not only IS risk assessment is not a well-researched area but also most research in IS risk assessment focus on IS security. However, our current focus is entirely different and we focus on risk in profit making of providers. So we make two major contributions here. We

use Mean Variance analysis model in IS research. We also introduce a new perspective of risk in IS literature.

5.2 Contributions to Practice

We obtained some results that we believe will be important for the practitioners.

We recognized the importance of non-functional attributes in the context of cloud computing applications. We identified modularity and performance as two important non-functional attributes. We identified that performance is a two dimensional attribute; it has two parts architectural and operational.

We uncovered the role of modularity and architectural performance in the architecture of a cloud computing application. We compared two cases where modularity and architectural performance are independent of each other and they are inversely related. We found that optimal values of each are independent of the fact whether there is a relationship between modularity and architectural performance.

We also found that practitioners need to plan for capacity that is greater than the optimal demand. We provided a way for the practitioners to calculate planned optimal capacity.

We provided an intuitive way to include risk tolerance in the model. We found out that a risk averse provider will have plan for a capacity that will be less than a risk taking provider.

Bibliography

- Agrawal, M., & Chari, K. (2007, March). Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 33(3), 145-156.
- Alberts, C. J., & Dorofee, A. J. (2001). *OCTAVE Criteria, Version 2.0*. Pittsburgh: Networked Systems Survivability Program, Software Engineering Institute, CMU.
- Alter, S., & Sherer, S. A. (2004). A General, But Readily Adaptable Model of Information System Risk. *Communications of AIS*, 14(2), 1-28.
- Amazon.com. (n.d.). *Amazon Elastic Compute Cloud (Amazon EC2)*. Retrieved December 26, 2011, from Amazon.com : <http://aws.amazon.com/ec2/>
- Arora, A., Hall, D., Pinto, C. A., Ramsey, D., & Telang, R. (2004). Measuring the Risk-Based Value of IT Security Solutions. *IT Pro*, 35-42.
- Bakos, Y., & Brynjolfsson, E. (1999). Bundling Information Goods: Pricing, Profits, and Efficiency. *Management Science*, 1613-1630.
- Baldwin, C. Y., & Clark, K. B. (2000). *Design Rules* (Vol. 1). The MIT Press.
- Balsamo, S., Di Marco, A., Inverardi, P., & Simeoni, M. (2004, May). Model-Based Performance Prediction in Software Development: A Survey. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 30(5), 295-310.
- Bandyopadhyay, K., Mykytyn, P. P., & Mykytyn, K. (1999). A framework for integrated risk management in information technology. *Management Decision*, 37(5), 437-448.
- Banker, R. D., Datar, S. M., Kemerer, C. F., & Zweig, D. (1993). Software Complexity and Maintenance Costs. *Communications of the ACM*, 36(11), 81-94.
- Bardhan, I. R., Demirkan, H., Kannan, P., Kauffman, R. J., & Sougstad, R. (2010). An Interdisciplinary Perspective on IT Services Management and Service Science. *Journal of Management Information Systems*, 13-64.
- Barua, A., Kriebel, C. H., & Mukhopadhyay, T. (1991, September). An Economic Analysis of Strategic Information Technology Investments. *MIS Quarterly*, 15(3), 313-331.
- Bashir, A., & Christin, N. (2008). Three Case Studies in Quantitative Information Risk Analysis. *Proceedings of the CERT/SEI Business Case Workshop*:

- Making the Business Case for Software Assurance* (pp. 77-86). Pittsburgh: CERT/SEI.
- Bask, A., Lipponen, M., Rajahonka, M., & Tinnila, M. (2010). The concept of modularity: diffusion from manufacturing to service production. *Journal of Manufacturing Technology Management*, 21(3), 355-375.
- Benlian, A., Hess, T., & Buxman, P. (2009). Drivers of SaaS - Adoption - An Empirical Study of Different Application Types. *Business & Information Systems Engineering*, 5, 357-369.
- Bitner, M. J., & Brown, S. W. (2006). The Evolution and Discovery of Services Science in the Business Schools. *COMMUNICATIONS OF THE ACM*, 49(7), 73-78.
- Boehm, B. W., & Sullivan, K. J. (2000). Software economics: a roadmap. *Proceedings of the conference on The future of Software engineering* (pp. 319-343). ACM Press.
- Bojanc, R., & Jerman-Blaz'ic, B. (2008). An economic modelling approach to information security risk management. *International Journal of Information Management*, 28, 413-422.
- Bolton, R. N., Smith, A. K., & Wagner, J. (2003). Striking the Right Balance Designing Service to Enhance Business-to-Business Relationships. *Journal of Service Research*, 271-291.
- Booch, G. (1991). *Object Oriented Design with Applications*. The Benjamin/Cummings Publishing Company, Inc.
- Booch, G. (1994). *Object-Oriented Analysis and Design*. Redwood City, California: Benjamin/Cummings.
- Bush, A. A., Tiwana, A., & Rai, A. (2010). Complementarities Between Product Design Modularity and IT Infrastructure Flexibility in IT-Enabled Supply Chains. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, 240-254.
- Cai, Y. (2006). *Modularity in Design: Formal Modeling and Automated Analysis*. PhD Thesis, University of Virginia, School of Engineering and Applied Science.
- Campagnolo, D., & Camuffo, A. (2010). The Concept of Modularity in Management Studies: A Literature Review. *International Journal of Management Reviews*, 259-283.

- Capra, E., Francalanci, C., & Merlo, F. (2008). An Empirical Study on the Relationship among Software Design Quality, Development Effort, and Governance in Open Source Projects. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 34(6), 765-782.
- Caralli, R. A., Stevens, J. F., Young, L. R., & Wilson, W. R. (2007). *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process*. Pittsburgh: CERT Program, SOFTWARE ENGINEERING INSTITUTE, CMU.
- Carr, N. (2005). The End of Corporate Computing. *MIT SLOAN Management Review*, 48(3), pp. 66-73.
- Chidamber, S. R., & Kemerer, C. F. (1994 йил June). A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6), 476-493.
- Choudhary, V. (2007). Comparison of Software Quality Under Perpetual Licensing and Software as a Service. *Journal of Management Information Systems*, 141-165.
- Choudhary, V. (2007). Software as a Service: Implications for Investment in Software Development. *Proceedings of the 40th Hawaii International Conference on System Sciences*. IEEE Computer Society.
- Chung, L., & Sampaio do Prado Leite, J. C. (2009). On Non-Functional Requirements in Software Engineering. In A. Borgida, V. Chaudhri, P. Giorgini, & E. Yu (Eds.), *Conceptual Modeling: Foundations and Applications, LNCS 5600* (pp. 363-379). Springer.
- Clark, D. D. (1982). *MODULARITY AND EFFICIENCY IN PROTOCOL IMPLEMENTATION*. MIT, MIT Laboratory for Computer Science. Internet Engineering Task Force.
- Conklin, W., White, G., Williams, D., Davis, R. L., & Cothren, C. (2010). *Principles of Computer Security: CompTIA Security+ and Beyond*. McGraw-Hill.
- Council, T. P. (n.d.). Retrieved 2011 йил 19-July from www.tpc.org
- Cowell, D. W. (1988). New Service Development. *Journal of Marketing Management*, 3(3), 266-312.
- Demirkan, H. (2008). The Servitisation of Processes, Architectures and Technologies. *The International Journal of Services Sciences*, 1(3/4), 197-205.

- Demirkan, H., Cheng, H. K., & Bandyopadhyay, S. (2010). Coordination Strategies in an SaaS Supply Chain. *Journal of Management Information Systems*, 26(4), 119-143.
- Demirkan, H., Kauffman, R. J., Vayghan, J. A., Fill, H.-G., Karagiannis, D., & Maglio, P. (2008). Service-oriented technology and management: Perspectives on research and practice for the coming decade. *Electronic Commerce Research and Applications*, 7, 356-376.
- Devaraj, E., Kumar, S., Kavi, T., & Kanth, K. R. (2011). Predicting the software performance during feasibility study. *IET Software*, 5(2), 201-215.
- Dewan, R., Jing, B., & Seidmann, A. (2003). Product Customization and Price Competition on the Internet. *Management Science*, 49(8), 1055-1070.
- Dewan, S., Shi, C., & Gurbaxani, V. (2007). Investigating the Risk-Return Relationship of Information Technology Investment: Firm-Level Empirical Analysis. *Management Science*, 53(12), 1829-1842.
- Diaz, J. C., & Dutt, A. (1992). Experiences with Line Ordered-Nested Block Preconditioning for Non symmetric Systems on the CM-2. *Proc. of International Conference on Computer Methods for Partial Differential Equations (IMACS – PDE 7)*. .
- Fichman, R. G. (2004). Going Beyond the Dominant Paradigm for Information Technology Innovation Research: Emerging Concepts and Methods. *Journal of the Association of Information Systems*, 5(8), 314-355.
- Fitzsimmons, J. A., & Fitzsimmons, M. J. (2004). *Service Management Operations Strategy Information Technology*. (4, Ed.) McGraw Hill Irwin.
- Fixson, S. K. (2003). *The Multiple Faces of Modularity - A Literature Analysis of a Product Concept for Assembled Hardware Products*. University of Michigan, Industrial and Operations Engineering. Ann Arbor: Sebastian Fixson.
- Fixson, S., & Clark, J. (2002). On the link between modularity and cost-a methodology to assess cost implications of product architecture differences, 2002. IEMC '02. 2002 , vol.1, no., pp. 131- 136 vol. *Engineering Management Conference. 1*, pp. 131-136. IEEE International.
- Focardi, S., & Jonas, C. (1998). *Risk Management: Framework, Methods, and Practice*. New Hope, Pennsylvania.: Frank J. Fabozzi Associates.
- Fouquet, M., Niedermayer, H., & Carle, G. (2009). Cloud computing for the masses. *U-NET '09 Proceedings of the 1st ACM workshop on User-*

- provided networking: challenges and opportunities* (pp. 31-36). Rome, Italy: ACM, New York, NY, USA.
- Gadrey, J. (2000). The Characterization of Goods and Services: An Alternative Approach", *Review of Income and Wealth*, *46*(3), 369-387.
- Google. (n.d.). *Google App Engine - Google Code*. Retrieved December 26, 2011, from Google App Engine: <http://code.google.com/appengine/>
- Gregory, P. (2010). *CISSP Guide to Security Essentials*. Boston: Course Technology.
- Hanmer, R. S., & Letourneau, J. P. (2003). A Best Practice for Performance Engineering. *Bell Labs Technical Journal*, *8*(3), 75-89.
- Heim, G. R., & Sinha, K. K. (2002). Service Process Configurations in Electronic Retailing: A Taxonomic Analysis of Electronic Food Retailers. *Production and Operations Management*, *11*(4), 54-74.
- Heim, G. R., & Sinha, K. K. (2005). Service product Configurations in Electronic Business-to-Consumer Operations. *Journal of Service Research*, *11*(4), 360-376.
- Hosangar, K., Krishnan, R., Chuang, J., & Choudhary, V. (2005). Pricing and Resource Allocation in Caching Services With Multiple Levels of Quality of Service. *Management Science*, *51*(12), 1844-1859.
- Hull, F. M. (2004). A Composite Model of Product Development Effectiveness: Application to Services. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, *51*(2), 162-172.
- IBM. (n.d.). Retrieved May 9, 2011, from <http://www.research.ibm.com/ssme/services.shtml>
- Jain, S., & Kannan, P. K. (2002). Pricing of Information Products on Online Servers: Issues, Models, and Analysis. *Management Science*, *48*(9), 1123-1142.
- Joglekar, N. R., & Rosenthal, S. R. (2003). Coordination of Design Supply Chains for Bundling Physical and Software Products. *The Journal of Product Innovation Management*, 374-390.
- Katsikas, S. (2009). Risk Management. In J. R. Vacca (Ed.), *Computer and Information Security Handbook* (pp. 606-625). Morgan Kaufmann.

- Kossmann, D., Kraska, T., & Loesing, S. (2010). An Evaluation of Alternative Architectures for Transaction Processing in the Cloud. *SIGMOD' 10* (pp. 579-590). Indianapolis, Indiana, USA.: ACM.
- Krishnan, V., & Ulrich, K. T. (2001). Product Development Decisions: A Review of the Literature. *Management Science*, *47*(1), 1-21.
- Krishnan, V., & Zhu, W. (2006). Designing a Family of Development-Intensive Products. *Management Science*, *52*(6), 813–825.
- Kumar, A. (2004). Mass Customization: Metrics and Modularity. *The International Journal of Flexible Manufacturing Systems*, *16*, 287–311.
- Laguna, M. (1998, November). Applying Robust Optimization to Capacity Expansion of one Location in Telecommunications with Demand Uncertainty. *Management Science*, *44*(11), S101-S110.
- Landoll, D. J. (2006). *The Security Risk Assessment Handbook*. Boca Raton, Florida: Auerbach Publications.
- Langlois, R. N., & Garzarelli, G. (2008). Of Hackers and Hairdressers: Modularity and the Organizational Economics of Open-source Collaboration. *Industry and Innovation*, *15*(2), 125-143.
- Lau-Antonio, K., Yam, R. C., & Tang, E. (2007). The impacts of product modularity on competitive capabilities and performance: An empirical study. *Int. J. Production Economics*(105), 1-20.
- Li, Y., & Lee, Y. (2010). Pricing peer-produced services: Quality, capacity, and competition issues. *European Journal of Operational Research*, 1658-1668.
- Linthicum, D. S. (2009). *Cloud Computing and SOA Convergence in Your Enterprise*. Addison-Wesley.
- MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. *Management Science*, *52*(7), 1015-1030.
- MacCormack, A., Verganti, R., & Iansiti, M. (2001). Developing Products on "Internet Time": The Anatomy of a Flexible Development Process. *Management Science*, *47*(1), 133-150.
- Maglio, P. P., & Spohrer, J. (2008). Fundamentals of service science. *Journal of the Academy of Marketing Science*, *36*, 18-20.

- Magnusson, P. R., Matthing, J., & Kristensson, P. (2003). Managing User Involvement in Service Innovation. *Journal of Service Research*, 111-124.
- Markowitz, H. (1959). *Portfolio Selection: Efficient Diversification of Investment*. New Haven: Yale University Press.
- Marsaglia, G. (2004, July). Evaluating the Normal Distribution. *Journal of Statistical Software*, 11(4), 1-11.
- MAS Research Roadmap Project. (2005). Research Directions for Service-Oriented Multiagent Systems. *IEEE INTERNET COMPUTING*, 65-70.
- McConnell, S. (2000, January / February). The Best Influences on Software Engineering. *IEEE Software*, 10-17.
- Mead, N. R., Allen, J. H., Conklin, W. A., Drommi, A., Harrison, J., Ingalsbe, J., et al. (2009). *Making the Business Case for Software Assurance*. Pittsburgh: CERT Program, CMU.
- Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing (Draft)*. Information Technology Laboratory, Computer Security Division. Gaithersburg: National Institute of Standards and Technology.
- Menasce, D. A., & Ngo, P. (2009). Understanding Cloud Computing: Experimentation and Capacity Planning. *Computer Measurement Group Conference, Dallas, TX, Dec. 7-11, 2009*. Dallas, TX.
- Menor, L. J., Tatikonda, M. V., & Sampson, S. E. (2002). New service development: areas for exploitation and exploration. *Journal of Operations Management*, 20, 135-157.
- Mens, T., & Tourwe, T. (2004). A Survey of Software Refactoring. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 30(2), 126-139.
- Miller, M. (2004). Probabilistic Risk Analysis and the concept of Bayesian Networks. *Invited Talk*. University of Bielefeld.
- Moorthy, K. S. (1993). Theoretical Modeling in Marketing. *Journal of Marketing*, 92-106.
- Nambisan, S., & Wilemon, D. (2000). Software Development and New product Development: Potentials for Cross-Domain Knowledge Sharing. *IEEE Transactions on Engineering Management*, 47(2), 211-221.
- National Security Telecommunications and Information Systems Security Committee. (1994). *National Training Standard for Information Systems*

Security Professionals NSTISSI No. 4011. Fort George G. Meade: National Security Agency.

NVD Common Vulnerability Scoring System Support v2. (n.d.). Retrieved October 25, 2010, from National Vulnerability Database CVSS Scoring: <http://nvd.nist.gov/cvss.cfm>

Papazoglou, M. P., & den Heuvel, W. V. (2005, NOVEMBER DECEMBER). Web Services Management: A Survey. *IEEE INTERNET COMPUTING*, 58-64.

Parasuraman, A., Zeithaml, V. A., & Malhotra, A. (2005). E-S-Qual A Multiple-Item Scale for Assessing Electronic Service Quality. *Journal of Service Research*, 213-233.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems. *Communications of the ACM*, 15(12), 1053-1058.

Parnas, D. L., Clements, P. C., & Weiss, D. M. (1985). The Modular Structure of Complex Systems. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, SE-11(3), 259-266.

Pekkarinen, S., & Ulkuniemi, P. (2008). Modularity in developing business services by platform approach. *The International Journal of Logistics Management*, 19(1), 84-103.

Peltier, T. R. (2005). *Information Security Risk Analysis*. Auerbach Publications.

Raju, J. S. (1995). Theoretical models of sales promotions: Contributions, limitations, and a future research agenda. *European Journal of Operational Research*, 1-17.

Rathmell, J. M. (1966). What is meant by Services. *Journal of Marketing*, 30, 32-33.

Rust, R. T., & Chung, T. S. (2006). Marketing Models of Service and Relationships. *Marketing Science*, 25(6), 560-580.

Salesforce.com. (n.d.). *Force.com Cloud Computing - Programmable User Interface*. Retrieved July 11, 2011, from Salesforce.com: <http://www.salesforce.com/platform/cloud-platform/programmable-ui.jsp>

Salesforce.com. (n.d.). *Sales Force Automation - Salesforce.com*. Retrieved December 26, 2011, from Sales Cloud: <http://www.salesforce.com/crm/sales-force-automation/>

- Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary firms. *MIS Quarterly*, 237-264.
- Scheuing, E. E. (1989). *New Product Management*. Merril Publishing.
- Schilling, M. A. (2000). Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity. *The Academy of Management Review*, 25(2), 312-334.
- Smunt, T. E. (1996, August). Rough Cut Capacity Planning in a Learning Environment. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, 43(3), 334-341.
- Sourceforge.net. (n.d.). *Metrics 1.3.6*. Retrieved December 27, 2011, from Metrics 1.3.6 - Getting started: <http://metrics.sourceforge.net/>
- Spohrer, J., Maglio, P. P., Bailey, J., & Gruhl, D. (2007, January). Steps Toward a Science of Service Systems. *Computer*, 49(7), 71-77.
- Sridhar, T. (2011, December 27). *Cloud Computing - A Primer - The Internet Protocol Journal, Volume 12, No.3 - Cisco Systems*. Retrieved from Cloud Computing - A Primer: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-3/123_cloud1.html
- Stoneburner, G., Goguen, A., & Feringa, A. (2002). *Risk Management Guide for Information Technology Systems*. Gaithersburg, MD 20899-8930: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.
- Susarla, A., Barua, A., & Whinston, A. B. (2009). A Transaction Cost Perspective of the "Software as a Service" Business Model. *Journal of Management Information Systems*, 26(2), 205-240.
- Susarla, A., Barua, A., & Whinston, A. B. (2010). Multitask Agency, Modular Architecture, and Task Disaggregation in SaaS. *Journal of Management Information Systems*, 26(4), 87-117.
- Taleb, N., Goldstein, D. G., & Spitznagel, M. W. (2009). The Six Mistakes Executives Make In Risk Management. *Harvard Business Review*, 78-81.
- Taudes, A., Feurstein, M., & Mild, A. (2000). Options Analysis of Software Platform Decisions: A Case Study. *MIS Quarterly*, 227-243.

- Transaction Processing Performance Council (TPC). (2012). *TPC Virtual Measurement Single System TPC-VMS*. Transaction Processing Performance Council.
- Transaction Processing Performance Council. (n.d.). *TPC-W - HomePage*. Retrieved from TPC: <http://www.tpc.org/tpcw/default.asp>
- Ueno, K., & Tatsubori, M. (2009, October-December). Early Capacity Testing of an Enterprise Service Bus. *International Journal of Web Services Research*, 4(4), 30-47.
- Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy*, 24(3), 419-440.
- Vaquero, L. M., Rodero-Merion, L., Caceres, J., & Lindner, M. (2009, January). A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, 39(1), 50-55.
- Vitasek, K. (2010, February). *Resources & Research: Glossary of Terms*. Retrieved May 22, 2010, from Council of Supply Change Management Professionals (CSCMP): <http://cscmp.org/digital/glossary/document.pdf>
- Wang, J., Chaudhury, A., & Rao, H. R. (2008). A Value-at-Risk Approach to Information Security Investment. *Information Systems Research*, 106-120.
- Wasserman, A. I. (1996, November). Toward a Discipline of Software Engineering. *IEEE Software*, 23-31.
- Weng, Z. K. (1999). Risk-pooling over demand uncertainty in the presence of product modularity. *Int. J. Production Economics*(105), 75-85.
- Whitman, M. E., & Mattord, H. J. (2009). *Principles of Information Security*. Boston: Course Technology.
- Wikipedia. (n.d.). *Assembly Line - Wikipedia*. Retrieved July 8, 2011, from Wikipedia Web site: http://en.wikipedia.org/wiki/Assembly_line
- Wikipedia. (n.d.). *Software as a service*. Retrieved November 18, 2007, from <http://en.wikipedia.org/wiki/SaaS>
- Wikipedia.org. (n.d.). *Risk*. Retrieved June 1, 2010, from Wikipedia: <http://en.wikipedia.org/wiki/Risk>
- Winston, W. L. (1993). *Operations Research: Applications and Algorithms* (3rd Edition ed.). Duxbury Press.

- Wu, J., Li, J., Wang, S., & Cheng, T. C. (2009). Mean–variance analysis of the newsvendor model with stockout cost. *Omega*, *37*, 724-730.
- Wu, L.-C., & Ong, C.-S. (2008). Management of information technology investment: A framework based on a Real Options and Mean–Variance theory perspective. *Technovation*, *28*, 122–134.
- Yoo, Y. (2010, June). Computing in Everyday Life: A Call for Research on Experiential Computing. *MIS Quarterly*, *34*(2), 213-231.
- Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Information Systems Research*, *21*(4), 724-735.
- Yourdon, Y., & Constantine, L. (1979). *Structured Design: Fundamentals of a discipline of computer program and systems design*. Prentice Hall.
- Yue, W. T., Çakanyıldırım, M., Ryu, Y. U., & Liu, D. (2007). Network externalities, layered protection and IT security risk management. *Decision Support Systems*, 1-16.
- Zhang, J., & Seidmann, A. (2010). Perpetual Versus Subscription Licensing Under Quality Uncertainty and Network Externality Effects. *Journal of Management Information Systems*, *27*(1), 39-68.
- Zhang, Z., Tan, Y., & Dey, D. (2009). Price competition with service level guarantee in web services. *Decision Support Systems*, *47*, 93-104.

Appendix

```

package omegaSimulation;
import java.util.Random;

/*****
*****
* This program was originally developed by Sedgewick
* Modified by Abhijit Dutt
* The approximation is accurate to absolute error less than 8 *
10(-16).
* Reference: Evaluating the Normal Distribution by George Marsaglia.
* http://www.jstatsoft.org/v11/a04/paper
*
*****
*****/

public class Gaussian {

    // return phi(x) = standard Gaussian pdf
    public static double phi(double x) {
        return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
    }

    // return phi(x, mu, sigma) = Gaussian pdf with mean mu and
    stddev sigma
    public static double phi(double x, double mu, double sigma) {
        return phi((x - mu) / sigma) / sigma;
    }

    // return Phi(z) = standard Gaussian cdf using Taylor
    approximation
    public static double Phi(double z) {
        if (z < -8.0) return 0.0;
        if (z > 8.0) return 1.0;
        double sum = 0.0, term = z;
        for (int i = 3; sum + term != sum; i += 2) {
            sum = sum + term;
            term = term * z * z / i;
        }
    }
}

```

```

        return 0.5 + sum * phi(z);
    }

    // return Phi(z, mu, sigma) = Gaussian cdf with mean mu and
    // stddev sigma
    public static double Phi(double z, double mu, double sigma) {
        return Phi((z - mu) / sigma);
    }

    // Compute z such that Phi(z) = y via bisection search
    public static double PhiInverse(double y) {
        return PhiInverse(y, .00000001, -8, 8);
    }

    // bisection search
    private static double PhiInverse(double y, double delta,
    double lo, double hi) {
        double mid = lo + (hi - lo) / 2;
        if (hi - lo < delta) return mid;
        if (Phi(mid) > y) return PhiInverse(y, delta, lo, mid);
        else return PhiInverse(y, delta, mid, hi);
    }

    // test client
}

```

```

/*****
*****
*   This program has been originally developed by Abhijit Dutt
*   This class is used for calculating the optimal parameters
*   when provider's risk tolerance is not included.
*
*
*****
*****/

```

```

package omegaSimulation;
import java.awt.Color;
import java.io.*;
import java.util.*;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.jfree.chart.plot.ValueMarker;
import org.jfree.chart.plot.XYPlot;

public class mySimulation {

    final double alpha = 100;
    final double beta = 3;
    protected double omega = .05;
    protected double delta = 1;
    protected double Z= 20;
    protected double A = 5;
    protected double D= 10;
    protected double demand = 0;
    protected double capacityPDF = 0;
    protected double price = 0;

```



```

protected double opportunity = 0;
protected double performance = 0;
protected double dataStore [][];
protected ChartHandler myChart = new ChartHandler("My
Chart");
protected double variancePercentage;

public mySimulation(double var, int numVariables, int
numRows){
    dataStore = new double [numVariables][numRows];

    variancePercentage= var;
    updateParameters();

}
public void updateParameters(){
    demand = 2*D*beta*(alpha - Z*beta)/(4*D*beta - delta*delta);
    price = 2*D*(alpha - Z*beta)/(4*D*beta - delta*delta) +Z;
    performance = delta*(alpha - Z*beta)/(4*D*beta -
delta*delta) ;
    opportunity = getPrice() -Z*omega;
    //opportunity = 0;
}
public double getDemand(){

    return (demand);
}
public double getCapacityPDF(){
    //capacityPDF = 1 - (Z*omega/(price+opportunity));
    capacityPDF = (price+opportunity- Z)/(price+opportunity-
Z*(1-omega) );
    return capacityPDF;
}

```

```

public double getPrice(){

    return price;
}
public double getPerformance(){

    return performance;
}

public double getOpportunity(){

    return opportunity;
}

public void printIt(){

    //System.out.printf("Demand ", demand);

    //System.out.printf("Demand: %03D", demand + " Price:" +
price+ " Opportunity:" + opportunity);
    System.out.println(String.format("Demand: %5.2f Price:
%5.2f Opportunity: %5.2f Performance: %5.2f", demand,
price, opportunity, performance));

}

public double calculateOptimalCapacity(double t_omega){
    omega = t_omega;
    updateParameters();
    double mu = getDemand();
    double sigma = variancePercentage*mu;

    double y = getCapacityPDF();

```

```

double phi_inv = Gaussian.PhiInverse(y)*sigma+mu;
return phi_inv;

}

public void omegaSimulation(double var, int num, int
numSteps){

    variancePercentage= var;
    omega = .35;
    updateParameters();
    for (int i=0; i < numSteps; i++){

        double phi_inv = calculateOptimalCapacity(omega);
        dataStore[0][i] = opportunity;
        dataStore [num][i] = phi_inv;
        //System.out.println( "Omega:"+simul.omega + "    y"+ y+ "
PHI-INVERSE:"+ phi_inv +"    Diff:"+ (mu-phi_inv));
        //System.out.println(String.format("Omega:%5.2f  PHI-
INVERSE:%5.2f  Diff:%5.2f", simul.omega,
phi_inv,(simul.demand-phi_inv)));
        //data.put(i, new Object[] {simul.omega, simul.demand,
phi_inv,(simul.demand-phi_inv)});
        omega += .05;
    }
}

public void opportunitySimulation(double var, int num, int
numSteps){

    variancePercentage= var;
    omega = .6;

```

```

updateParameters();
opportunity = 0;
Map<Object, Object[]> data = new TreeMap<Object,
Object[]>();

double incr = (getPrice() -Z*omega)*1.5/numSteps;
for (int i=0; i < numSteps; i++){

double phi_inv = calculateOptimalCapacity(omega);
dataStore[0][i] = opportunity;
dataStore [num][i] = phi_inv;
//System.out.println( "Omega:"+simul.omega + "  y"+ y+ "
PHI-INVERSE:"+ phi_inv +"  Diff:"+ (mu-phi_inv));
//System.out.println(String.format("Omega:%5.2f  PHI-
INVERSE:%5.2f  Diff:%5.2f", simul.omega,
phi_inv,(simul.demand-phi_inv));
//data.put(i, new Object[] {simul.omega, simul.demand,
phi_inv,(simul.demand-phi_inv)});
opportunity += incr;
}
for (int i=0; i < numSteps; i++){

//System.out.println( "Omega:"+simul.omega + "
y"+ y+ "  PHI-INVERSE:"+ phi_inv +"  Diff:"+ (mu-
phi_inv));
//System.out.println(String.format("Omega:%5.2f  PHI-
INVERSE:%5.2f  Diff:%5.2f", simul.omega,
phi_inv,(simul.demand-phi_inv));
data.put(i, new Object[] {dataStore[0][i],
dataStore[1][i]});

}
//excelWriter temp = new excelWriter();

```

```

        //temp.writeFile(data,
"C:\\Users\\aud18\\Documents\\research\\Dissertation\\Paper2\\
\\test.xls");
drawChart(data, "Opportunity", 1, new String[] {"Capacity"});
myChart.drawHorizontalLine(demand);
myChart.setVisible(true);

    }
public static void main(String[] args) {
    int numOfSteps = 15;
    mySimulation simul = new mySimulation(.1, 4, numOfSteps);
    /*
Map<Object, Object[]> data = new TreeMap<Object, Object[]>();

    simul.printIt();
    simul.omegaSimulation(.01, 1, numOfSteps);
    simul.omegaSimulation(.05, 2, numOfSteps);
    simul.omegaSimulation(.1, 3, numOfSteps);

        //data.put(0, new Object[] {"Omega", "y", "Phi-Inv",
"Diff"});
        for (int i=0; i < numOfSteps; i++){

                //System.out.println( "Omega:"+simul.omega + "
y"+ y+ "      PHI-INVERSE:"+ phi_inv + "   Diff:"+ (mu-
phi_inv));
                //System.out.println(String.format("Omega:%5.2f
PHI-INVERSE:%5.2f   Diff:%5.2f", simul.omega,
phi_inv,(simul.demand-phi_inv)));
                data.put(i, new Object[] {simul.dataStore[0][i],
simul.dataStore[1][i], simul.dataStore[2][i],
simul.dataStore[3][i]});

```

```

    }
    //excelWriter temp = new excelWriter();
    //temp.writeFile(data,
"C:\\Users\\aud18\\Documents\\research\\Dissertation\\Paper2\\
\\test.xls");
        simul.drawChart(data, "Omega", 3, new String[]
{"Capacity(Var = .01)", "Capacity(Var = .05)", "Capacity (Var
= .1)"});
        simul.myChart.drawHorizontalLine(simul.demand);
        simul.myChart.setVisible(true);

        */
simul.printIt();
simul.opportunitySimulation(.01, 1, numOfSteps);

}

```

```

public void drawChart(Map <Object, Object[]> inData, String
cName, int numDepVariab, String [] seriesName){
    Map<Object, Object[]> data;
    data = inData;

    Set<Object> keyset = data.keySet();

    int rowSize = keyset.size();
    //Number of rows
    int colSize = numDepVariab;

    System.out.println("Row:" + rowSize + " Col Size:" + colSize );

    double [] xValues= new double[rowSize];
    double [][] yValues= new double[colSize][rowSize];

```

```
for (Object key : keyset) {
    Object [] objArr = data.get(key);
    xValues[(Integer)(key)] = (Double)objArr[0];
    for (int j=1; j < (colSize+1); j++){
        yValues[j-1][(Integer)(key)] = (Double)objArr[j];
    }
}

//myChart.populateDataSet("Y", xValues, yValues[0]);
for (int i = 0; i < numDepVariab; i++)
    myChart.populateDataSet(seriesName[i], xValues,
yValues[i]);

myChart.drawChart(cName);

}

}
```

```

/*****
*****
* This program has been originally developed by Abhijit Dutt
* This class is used for performing numerical simulation when
* risk is ignored.
*

*****/
package ProfitSimulation;
import java.awt.Color;
import java.util.Map;
import java.util.Random;
import java.util.TreeMap;

import omegaSimulation.*;

public class profitAverage extends mySimulation{

    long totalZeroDemand = 0;

    double fixedCost = 0;
    double optimalCapacity = 0;

    double currentCapacity = 0;
    public final int numTrials=100000;

    private double sumOfProfit=0;

    private final Random rand = new Random();

    double averageDemandSum =0;

    public profitAverage(double vPerc){
        super(vPerc,10, 10);
        printIt();
    }
    //Given Random demand calculate profit
    public double calculateProfit (double r_demand){
        double r_profit = price*Math.min(currentCapacity,
r_demand)
                -Z*omega*currentCapacity -
Math.min(currentCapacity, r_demand)*Z*(1-omega)-
(A+D*performance*performance)

```



```

        - opportunity *Math.max((r_demand-
currentCapacity), 0);

        return r_profit;
    }
    public double calculateProfitWithoutOpportunity (double
r_demand){
        double r_profit = price*Math.min(currentCapacity,
r_demand)
            -Z*omega*currentCapacity -
Math.min(currentCapacity, r_demand)*Z*(1-omega)-
(A+D*performance*performance);

        return r_profit;
    }
    /**
     * @param args
     */

    public void simulationCapacity(){

        /*Check the logic */
        //First calculate the capacity, based on all
information
        //Keep increasing the capacity and check its effect on
average
        //demand
        //Use the average demand and and generate random
demand

        double avProfit2[]= new double[numTrials];
        long k = 10;
        double sd =0;
        Map<Object, Object[]> data = new TreeMap<Object,
Object[]>();
        omega = .3;
        updateParameters();
        optimalCapacity = calculateOptimalCapacity(omega);
        System.out.printf("Optimal Capacity:%5.2f
\n",optimalCapacity);
        double
increment=(2*variancePercentage*demand)*Math.abs(optimalCapacity-
demand)/k;
        //Update Current capacity and it is used for
calculating profit
        currentCapacity = optimalCapacity-(k/2)*increment;

```

```

for (int i=0; i < k; i++){
sumOfProfit=0;
double sumOfProfit2=0;
totalZeroDemand = 0;
currentCapacity += increment;
averageDemandSum =0;
long numOfEvents = 0;
for (int j =0; j< numTrials;j++){
double randomDemandComp =
rand.nextGaussian()*(variancePercentage*demand);
double randDemand = demand+
randomDemandComp;
if (randDemand > 0){
double randProfit =
calculateProfit(randDemand);
sumOfProfit += randProfit;
averageDemandSum +=randDemand;
randProfit =
calculateProfitWithoutOpportunity(randDemand);
avProfit2[j] = randProfit;

numOfEvents++;

}

}
//System.out.println( "Omega:"+simul.omega + "
y"+ y+ " PHI-INVERSE:"+ phi_inv + " Diff:"+ (mu-phi_inv));
//System.out.println(String.format("Omega:%5.2f
PHI-INVERSE:%5.2f Diff:%5.2f", omega, phi_inv,(demand-
phi_inv));
for(int n= 0 ; n<numOfEvents; n++)
sumOfProfit2 += avProfit2[n];

double averageProfit = sumOfProfit2/numOfEvents;
double sdSum1= 0;
for(int n= 0 ; n<numOfEvents; n++){
double temp2 = avProfit2[n]-averageProfit;
sdSum1 += temp2*temp2;
}
sd = Math.pow((sdSum1/(numOfEvents-1)),
.5);
averageProfit = sumOfProfit/numOfEvents;

```

```

        data.put(i, new Object[]
{currentCapacity,averageProfit, sd});

        //System.out.println("Average Profit:"+
(sumOfProfit/numTrials)+" Omega:"+omega+ "current cap:"+
currentCapacity+"Average Demand:"+ (averageDemandSum/numTrials));
        System.out.println(String.format("Average Profit:
%5.2f Current cap: %5.2f Average Demand:%5.2f SD:%5.2f",
(sumOfProfit/numTrials),currentCapacity,(averageDemandSum/numTria
ls), sd));
    }
    drawChart(data, "Capacity", 2, new String []
{"Average Profit", "SD Profit"});
    //drawChart(data, "Capacity", 1, new String []
{"Average Profit"});

    myChart.drawVerticalLine(optimalCapacity);

    //myChart.drawVerticalLine(demand);
    //myChart.drawHorizontalLine(calculateProfit(demand),
Color.green);
    myChart.setVisible(true);
}

/*
public void simulationVariation(){
    //First calculate the capacity, based on all
information
    //Keep increasing the capacity and check its effect on
average
    //demand
    //Use the average demand and and generate random
demand

    long k = 20;
    Map<Object, Object[]> data = new TreeMap<Object,
Object[]>();
    omega = .3;
    variancePercentage = .01;
    updateParameters();
    optimalCapacity = calculateOptimalCapacity(omega);
    System.out.printf("Optimal Capacity:%5.2f
\n",optimalCapacity);
    double
increment=(2*variancePercentage*demand)*Math.abs(optimalCapacity-
demand)/k;
    currentCapacity = optimalCapacity-(k/2)*increment;

```

```

        for (int i=0; i < k; i++){
            optimalCapacity =
calculateOptimalCapacity(omega);
            data.put(i, new Object[] {variancePercentage,
optimalCapacity});
            variancePercentage +=.01;
            //System.out.println("Average Profit:"+
(sumOfProfit/numTrials)+" Omega:"+omega+ "current cap:"+
currentCapacity+"Average Demand:"+ (averageDemandSum/numTrials));
            //System.out.println(String.format("Average
Profit: %5.2f Current cap: %5.2f Average Demand:%5.2f",
(sumOfProfit/numTrials),currentCapacity,(averageDemandSum/numTria
ls)));
        }
        drawChart(data, "Variance", 1, new String []
{"OptimalCapacity"});
        myChart.drawHorizontalLine(demand);

        myChart.setVisible(true);
    }
    */

    public void simulationCapacityWithVariance(){
        /*Check the logic */
        //First calculate the capacity, based on all
information
        //Keep increasing the capacity and check its effect on
average
        //demand
        //Use the average demand and and generate random
demand

        double avProfit2[]= new double[numTrials];
        long k = 100;
        double sd =0;
        Map<Object, Object[]> data = new TreeMap<Object,
Object[]>();
        omega = .3;
        updateParameters();
        optimalCapacity = calculateOptimalCapacity(omega);
        System.out.printf("Optimal Capacity:%5.2f
\n",optimalCapacity);
        double
increment=(2*variancePercentage*demand)*Math.abs(optimalCapacity-
demand)/k;
        //Update Current capacity and it is used for
calculating profit

```

```

currentCapacity = optimalCapacity-(k/2)*increment;
double highestProfitWithVar = 0;
double varOptCapacity = 0;
for (int i=0; i < k; i++){
    sumOfProfit=0;
    double sumOfProfit2=0;
    totalZeroDemand = 0;
    currentCapacity += increment;
    averageDemandSum =0;
    long numOfEvents = 0;
    for (int j =0; j< numTrials;j++){
        double randomDemandComp =
rand.nextGaussian()*(variancePercentage*demand);
        double randDemand = demand+
randomDemandComp;
        if (randDemand> 0){
            double randProfit =
calculateProfit(randDemand);
            sumOfProfit += randProfit;
            averageDemandSum +=randDemand;
            //randProfit =
calculateProfitWithoutOpportunity(randDemand);
            avProfit2[j] = randProfit;

            numOfEvents++;
        }
    }
}

//System.out.println( "Omega:"+simul.omega + "
y"+ y+ " PHI-INVERSE:"+ phi_inv + " Diff:"+ (mu-phi_inv));
//System.out.println(String.format("Omega:%5.2f
PHI-INVERSE:%5.2f Diff:%5.2f", omega, phi_inv,(demand-
phi_inv));
for(int n= 0 ; n<numOfEvents; n++)
    sumOfProfit2 += avProfit2[n];

double averageProfit = sumOfProfit2/numOfEvents;
double sdSum1= 0;
for(int n= 0 ; n<numOfEvents; n++){
    double temp2 = avProfit2[n]-averageProfit;
    sdSum1 += temp2*temp2;
}

```

```

    }
        sd = Math.pow((sdSum1/(numOfEvents-1)),
.5);
        averageProfit = sumOfProfit/numOfEvents;
        if (highestProfitWithVar < (averageProfit-sd)){
            highestProfitWithVar=averageProfit-sd;
            varOptCapacity = currentCapacity;
        }
        data.put(i, new Object[]
{currentCapacity,averageProfit, averageProfit-sd, sd});

        //System.out.println("Average Profit:"+
(sumOfProfit/numTrials)+" Omega:"+omega+ "current cap:"+
currentCapacity+"Average Demand:"+ (averageDemandSum/numTrials));
        System.out.println(String.format("Average Profit:
%5.2f Current cap: %5.2f Average Demand:%5.2f SD:%5.2f",
(sumOfProfit/numTrials),currentCapacity,(averageDemandSum/numTria
ls), sd));
    }
        drawChart(data, "Capacity", 3, new String []
{"Average Profit", "Av Profit2", "SD Profit"});
        //drawChart(data, "Capacity", 1, new String []
{"Average Profit"});

        myChart.drawVerticalLine(optimalCapacity);
        myChart.drawVerticalLine(varOptCapacity);

        //myChart.drawVerticalLine(demand);
        //myChart.drawHorizontalLine(calculateProfit(demand),
Color.green);
        myChart.setVisible(true);

    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //new profitAverage(.1).simulationCapacity();
        new
profitAverage(.15).simulationCapacityWithVariance();
    }
}

```

```

/*****
*****
*   This program has been originally developed by Abhijit Dutt
*   This class is used for calculating the optimal parameters
*   when provider's risk tolerance is included.
*
*
*****
*****/

package omegaSimulation;
import java.awt.Color;
import java.io.*;
import java.util.*;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.jfree.chart.plot.ValueMarker;
import org.jfree.chart.plot.XYPlot;

public class mySimulation {

    final double alpha = 100;
    final double beta = 1;
    final double gamma = 1;
    protected double omega = .05;
    protected double delta = 5;
    protected double Z= 50;
    protected double A = 5;
    protected double B = 5;
    protected double C = 5;
    protected double D= 10;
    protected double G= 3;

```

```

protected double demand = 0;
protected double capacityPDF = 0;
protected double price = 0;
protected double modularity = 0;
protected double opportunity = 0;
protected double performance = 0;
protected double dataStore [][];
protected ChartHandler myChart = new ChartHandler("My
Chart");
protected double variancePercentage;

public mySimulation(double var, int numVariables, int
numRows){
    dataStore = new double [numVariables][numRows];

    variancePercentage= var;
    updateParameters();

}

public void updateParameters(){

    double denom = (4*C*D*beta - D*gamma*gamma - C *
Math.pow((delta- G * beta), 2)) ;

    demand = (D*beta*(2*C*(alpha - Z*beta)+ B * gamma))/denom ;

    performance = (2*C*(alpha - Z*beta)+ B * gamma)*(delta-G *
beta)/(2*denom) ;

    price = D*(2*C*(alpha - Z*beta)+ B * gamma)/denom +Z +
G*performance;

    modularity = (2*D*(alpha - Z*beta)*gamma + B *(4*D*beta -
Math.pow((delta-G * beta), 2)))/(2*denom);

```



```

    opportunity = (getPrice() -Z);

    System.out.println("Hessian:" + denom);
    //opportunity = 0;
}
public double getDemand(){

    return (demand);
}
public double getCapacityPDF(){
    //capacityPDF = 1 - (Z*omega/(price+opportunity));
    capacityPDF = (price+opportunity- Z -
G*performance)/(price+opportunity- Z*(1-omega) );
    return capacityPDF;
}
public double getPrice(){

    return price;
}
public double getPerformance(){

    return performance;
}

public double getOpportunity(){

    return opportunity;
}

public void printIt(){

```

```

//System.out.printf("Demand ", demand);

//System.out.printf("Demand: %03D", demand + " Price:" +
price+ " Opportunity:" + opportunity);
System.out.println(String.format("Demand: %5.2f Price:
%5.2f Opportunity: %5.2f Performance: %5.2f Modularity:
%5.2f", demand, price, opportunity, performance,
modularity));

}

public double calculateOptimalCapacity(double t_omega){
    omega = t_omega;
    updateParameters();
    double mu = getDemand();
    double sigma = variancePercentage*mu;

    double y = getCapacityPDF();
    double phi_inv = Gaussian.PhiInverse(y)*sigma+mu;
    return phi_inv;
}

public void omegaSimulation(double var, int num, int
numSteps){

    variancePercentage= var;
    omega = .35;
    updateParameters();
    for (int i=0; i < numSteps; i++){

        double phi_inv = calculateOptimalCapacity(omega);

```

```

    datastore[0][i] = opportunity;
    datastore [num][i] = phi_inv;
    //System.out.println( "Omega:"+simul.omega + "    y"+ y+ "
    PHI-INVERSE:"+ phi_inv +"    Diff:"+ (mu-phi_inv));
    //System.out.println(String.format("Omega:%5.2f  PHI-
    INVERSE:%5.2f  Diff:%5.2f", simul.omega,
    phi_inv,(simul.demand-phi_inv)));
    //data.put(i, new Object[] {simul.omega, simul.demand,
    phi_inv,(simul.demand-phi_inv)});
    omega += .05;
}
}

public void opportunitySimulation(double var, int num, int
numSteps){

    variancePercentage= var;
    omega = .6;
    updateParameters();
    opportunity = 0;
    Map<Object, Object[]> data = new TreeMap<Object,
Object[]>();

    double incr = (getPrice() -Z*omega)*1.5/numSteps;
    for (int i=0; i < numSteps; i++){

        double phi_inv = calculateOptimalCapacity(omega);
        datastore[0][i] = opportunity;
        datastore [num][i] = phi_inv;
        //System.out.println( "Omega:"+simul.omega + "    y"+ y+ "
        PHI-INVERSE:"+ phi_inv +"    Diff:"+ (mu-phi_inv));
        //System.out.println(String.format("Omega:%5.2f  PHI-
        INVERSE:%5.2f  Diff:%5.2f", simul.omega,
        phi_inv,(simul.demand-phi_inv)));

```

```

    //data.put(i, new Object[] {simul.omega, simul.demand,
    phi_inv,(simul.demand-phi_inv)});
    opportunity += incr;
}
for (int i=0; i < numSteps; i++){

        //System.out.println( "Omega:"+simul.omega + "
y"+ y+ "    PHI-INVERSE:"+ phi_inv +"    Diff:"+ (mu-
phi_inv));
    //System.out.println(String.format("Omega:%5.2f  PHI-
INVERSE:%5.2f  Diff:%5.2f", simul.omega,
phi_inv,(simul.demand-phi_inv)));
    data.put(i, new Object[] {dataStore[0][i],
dataStore[1][i]});

}

    //excelWriter temp = new excelWriter();
    //temp.writeFile(data,
"C:\\Users\\aud18\\Documents\\research\\Dissertation\\Paper2\\
\\test.xls");
drawChart(data, "Opportunity", 1, new String[] {"Capacity"});
myChart.drawHorizontalLine(demand);
myChart.setVisible(true);

}

public static void main(String[] args) {
    int numOfSteps = 15;
    mySimulation simul = new mySimulation(.1, 4, numOfSteps);
    /*
    Map<Object, Object[]> data = new TreeMap<Object, Object[]>();

    simul.printIt();
    simul.omegaSimulation(.01, 1, numOfSteps);

```

```

simul.omegaSimulation(.05, 2, numOfSteps);
simul.omegaSimulation(.1, 3, numOfSteps);

    //data.put(0, new Object[] {"Omega", "y", "Phi-Inv",
"Diff"});
    for (int i=0; i < numOfSteps; i++){

        //System.out.println( "Omega:"+simul.omega + "
y"+ y+ "    PHI-INVERSE:"+ phi_inv + "    Diff:"+ (mu-
phi_inv));
        //System.out.println(String.format("Omega:%5.2f
PHI-INVERSE:%5.2f    Diff:%5.2f", simul.omega,
phi_inv,(simul.demand-phi_inv));
        data.put(i, new Object[] {simul.dataStore[0][i],
simul.dataStore[1][i], simul.dataStore[2][i],
simul.dataStore[3][i]});

    }

    //excelWriter temp = new excelWriter();
    //temp.writeFile(data,
"C:\\Users\\aud18\\Documents\\research\\Dissertation\\Paper2\\
\\test.xls");

    simul.drawChart(data, "Omega", 3, new String[]
{"Capacity(Var = .01)", "Capacity(Var = .05)", "Capacity (Var
= .1)"});

    simul.myChart.drawHorizontalLine(simul.demand);
    simul.myChart.setVisible(true);

    */
simul.printIt();
//simul.opportunitySimulation(.01, 1, numOfSteps);

}

```

```

public void drawChart(Map <Object, Object[]> inData, String
    cName, int numDepVariab, String [] seriesName){
    Map<Object, Object[]> data;
    data = inData;

    Set<Object> keyset = data.keySet();
    int rowSize = keyset.size();
    //Number of rows
    int colSize = numDepVariab;
    System.out.println("Row:"+ rowSize+" Col Size:"+ colSize );

    double [] xValues= new double[rowSize];
    double [][] yValues= new double[colSize][rowSize];

    for (Object key : keyset) {
        Object [] objArr = data.get(key);
        xValues[(Integer)(key)] = (Double)objArr[0];
        for (int j=1; j < (colSize+1); j++){
            yValues[j-1][(Integer)(key)] = (Double)objArr[j];
        }
    }

    //myChart.populateDataSet("Y", xValues, yValues[0]);
    for (int i = 0; i < numDepVariab; i++)
        myChart.populateDataSet(seriesName[i], xValues,
yValues[i]);

    myChart.drawChart(cName);

}

}

```

```

/*****
*****
*   This program has been originally developed by Abhijit Dutt
*   This class is used for performing numerical simulation when
*   risk is ignored.
*
*****
*****/

package ProfitSimulation;
import java.awt.Color;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.math.BigDecimal;
import java.math.MathContext;
import java.util.Date;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.TreeMap;

import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import omegaSimulation.*;

public class profitAverage extends mySimulation{

    long totalZeroDemand = 0;

```

```

double fixedCost = 0;
double optimalCapacity = 0;

double currentCapacity = 0;
//public final int numTrials=100000;
public final int numTrials=10000;

private final Random rand = new Random(100000);

double averageDemandSum =0;

public profitAverage(double vPerc){
    super(vPerc,10, 10);
}

//Given Random demand calculate profit
public double calculateProfit (double r_demand){

    //Use Equation 13
    double r_profit = price*Math.min(currentCapacity, r_demand)
        - (Z*omega+G*performance)*currentCapacity
        - Math.min(currentCapacity, r_demand)*Z*(1-omega)
        - (A -
B*modularity+C*modularity*modularity+D*performance*performanc
e)
        - opportunity *Math.max((r_demand-
currentCapacity), 0);
    return r_profit;
}

public double calculateProfitWithoutOpportunity (double
r_demand){

```



```

double r_profit = price*Math.min(currentCapacity, r_demand)
                -(Z*omega+G*performance)*currentCapacity
                - Math.min(currentCapacity, r_demand)*Z*(1-omega)
                -(A -
B*modularity+C*modularity*modularity+D*performance*performanc
e);

return r_profit;
}
/**
 * @param args
 */

public void simulationCapacityWithVariance(){
    /*Check the logic */
    //First calculate the capacity, based on all information
    //Keep increasing the capacity and check its effect on
    average
    //demand
    //Use the average demand and and generate random demand
    //

    //long k = 10000;

    long k = 5000;

    HSSFWorkbook workbook = new HSSFWorkbook();
    omega = .1;

    for (int pp=0; pp <5; pp++){
        omega += .1d;

```

```

        HSSFSSheet sheet = workbook.createSheet("Omega="+
Double.toString(omega).substring(0,3));

        updateParameters();
        printIt();

        optimalCapacity =
calculateOptimalCapacity(omega);
        System.out.printf("Optimal Capacity:%5.2f
demand:%5.2f \n",optimalCapacity, demand);
        double
increment=(variancePercentage*demand)*Math.abs(optimalCapacit
y-demand)/k;
        //Update Current capacity and it is used for
calculating profit
        System.out.printf("Start Capacity:%5.2f      End
Capacity::%5.2f incr:%4.3f \n",(optimalCapacity-
(k/2)*increment), (optimalCapacity+(k/2)*increment),
increment);

        double var =0;
        double epsilon = 0;
        Map<Integer, Object[]> data = new
TreeMap<Integer, Object[]>();

        int myKey = 1;
        data.put(myKey++, new Object[] {"Epsilon",
"Optimal Capacity", "H-Prof", "Av-Prof1", "Av-Prof2",
"SD"});

        for (int m=0; m < 10; m++){//Change Epsilon
            double highestProfitWithVar = 0;

```

```

double varOptCapacity = 0;
double myAverageProfit = 0;
double myAverageProfit2 = 0;
double mySD = 0;
currentCapacity = optimalCapacity-
(k/2)*increment;
epsilon += .1;

for (int i=0; i < k; i++){
    //This loop is for changing the capacity
    BigDecimal sumOfProfit= new BigDecimal(0.0,
MathContext.DECIMAL128);

    BigDecimal sumOfProfit2= new
BigDecimal(0.0, MathContext.DECIMAL128);

    totalZeroDemand = 0;
    currentCapacity += increment;
    averageDemandSum =0;
    long numOfEvents = 0;
    for (int j =0; j< numTrials;j++){
        double randomDemandComp =
rand.nextGaussian()*(variancePercentage*demand);
        double randDemand = demand+
randomDemandComp;

        if (randDemand> 0){
            double randProfit =
calculateProfit(randDemand);
            averageDemandSum +=randDemand;
            //randProfit =
calculateProfitWithoutOpportunity(randDemand);
            sumOfProfit=sumOfProfit.add(new
BigDecimal(randProfit), MathContext.DECIMAL128);

```

```

sumOfProfit2=sumOfProfit2.add(new
BigDecimal(randProfit*randProfit), MathContext.DECIMAL128);

                                numOfEvents++;

                                }

                                } //end for

                                //System.out.println( "Omega:"+simul.omega
+ "    y"+ y+ "    PHI-INVERSE:"+ phi_inv + "    Diff:"+ (mu-
phi_inv));

                                //System.out.println(String.format("Omega:%5.2f  PHI-
INVERSE:%5.2f  Diff:%5.2f", omega, phi_inv,(demand-
phi_inv)));

                                double averageProfit =
sumOfProfit.divide(new
BigDecimal(numOfEvents,MathContext.DECIMAL128),
MathContext.DECIMAL128).doubleValue() ;

                                var =
sumOfProfit2.subtract(sumOfProfit.multiply(new
BigDecimal(2*averageProfit))).doubleValue()/numOfEvents
+averageProfit*averageProfit;

                                double sd = Math.pow(var, 0.5);

                                double optimizationParam = averageProfit-
epsilon*sd;

                                if (highestProfitWithVar <
optimizationParam){

```

```

highestProfitWithVar=optimizationParam;
        varOptCapacity = currentCapacity;
        myAverageProfit2 = averageProfit;

    }
    if( myAverageProfit < averageProfit){
        myAverageProfit = averageProfit;
        mySD = sd;
    }
}

System.out.println(String.format("E:%5.2f
Opt Cap:%6.3f  H-Prof:%5.2f  Av-Prof1:%5.2f  Av-Prof2:%5.2f
SD :%5.2f",  epsilon, varOptCapacity,highestProfitWithVar,
myAverageProfit, myAverageProfit2,mySD));

        data.put(myKey++, new Object[] {epsilon,
varOptCapacity,highestProfitWithVar,  myAverageProfit,
myAverageProfit2,mySD});

} // Change Epsilon

Set<Integer> keyset = data.keySet();
int rownum = 0;
for (Integer key : keyset) {
    System.out.println("Key:"+key);
    Row row = sheet.createRow(rownum++);
    Object [] objArr = data.get(key);
    int cellnum = 0;
    for (Object obj : objArr) {
        Cell cell = row.createCell(cellnum++);
        if(obj instanceof Date)
            cell.setCellValue((Date)obj);
        else if(obj instanceof Boolean)
            cell.setCellValue((Boolean)obj);
    }
}

```

```

        else if(obj instanceof String)
            cell.setCellValue((String)obj);
        else if(obj instanceof Double)
            cell.setCellValue((Double)obj);
    }
}
}
try {
    FileOutputStream out =
        new FileOutputStream(new
File("C:\\Users\\aud18\\Documents\\research\\Dissertation\\Pa
per3\\Excel\\test.xls"));
    workbook.write(out);
    out.close();
    System.out.println("Excel written successfully..");

} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

}

public static void main(String[] args) {
    // TODO Auto-generated method stub
    //new profitAverage(.1).simulationCapacity();
    new profitAverage(.1).simulationCapacityWithVariance();
}

}

```

ABHIJIT DUTT

Office: (724) 773 3860

e mail: adutt77@yahoo.com

RESEARCH INTEREST:

- I am interested in the phenomenon of cloud computing. I consider cloud computing a significant step from the point of view of users. Although, from the point of view of technology it may not be very significant. In order to understand cloud computing, it is necessary to look into some other areas and use some innovative research methodology.
- Teaching different courses in the Security and Risk Analysis area and working in a NSF grant got me interested in this area. The third essay of my dissertation introduces innovative mean variance analysis theory to IS area. I would like to extend that to information assurance area. I also see that data mining could be a very important tool in the information assurance area. I would like to investigate that further.

EDUCATION: PhD Management Information Systems (major), Production and Operations Management (minor),

Univ. of Wisconsin-Milwaukee (**August, 2013**). Dissertation defended – June 21, 2013.

Dissertation Title: Economic Perspective on Cloud Computing: Three Essays.

MS in Computer Science, University of Tulsa, 1992.

MS in Physics, University of Calcutta, INDIA, 1983.

ACADEMIC PERPARATION:

UNIVERSITY OF WISCONSIN, MILWAUKEE, WI

Aug. '03 – Aug. '13

Ph.D. candidate, MIS Area, Lubar School of Business

- My dissertation examines issues faced by providers of Cloud Computing Applications. The first essay examines issues during development of SaaS. The second essay recognizes the service aspect of cloud computing and a model is developed to help providers with infrastructure capacity planning. The third essay examines the financial risks involved associated with capacity planning and a model is developed that would help providers to minimize their financial risk based on their risk tolerance. The primary research methodology used – theoretical modeling. Also used numerical solution and computer simulation. Custom Java programs were developed for the purpose.
- During my doctoral program I took many research methodology courses as well as Information System courses.

ACADEMIC EXPERIENCE:

PENN STATE UNIVERSITY, MONACA, PA

Aug. '08 – Till Date

Instructor of Information Sciences and Technology

- ✓ Responsible for teaching courses in the areas of Information Science and Technology and Security and Risk Analysis in Beaver campus as well in World campus of Penn State.
- ✓ Teach courses using variety of formats – traditional face to face in class, synchronous video conferencing and online. I have developed many of the Security and Risk Analysis courses.
- ✓ Also responsible for internship and advising students. I regularly organize trips for students to local companies. During my time in Penn State I have built strong relationships with many local companies. I also participate in student recruitment and serve in different committees.
- ✓ Advisor for Security and Risk Analysis minor. Investigator in a NSF grant “Exploration of a Collaborative Virtual Computer Laboratory (CVCLAB) to Enhance Distance Learning in Information Security”. Worked with other investigators and designed courses and virtual laboratories for online course offering.

Carnegie Mellon University, Pittsburgh, Pa
Adjunct Instructor

Jan. '10 – Till Date

I have taught three graduate courses in the Heinz College. I am scheduled to teach again in fall '13.

EDGEWOOD COLLEGE, MADISON, WI
Visiting Assistant Professor of Computer Information Systems

Aug. '07 – May '08

Taught **Networking Fundamentals** and **Network Security and Network Management** to returning adult students and **System Analysis & Design and Introduction to Information Systems** to traditional students. Was also involved in service activities such as student advising, hiring of adjunct faculty etc.

UNIVERSITY OF WISCONSIN, MILWAUKEE, WI
Adjunct instructor, MIS Area, Lubar School of Business

Aug. '03 – Aug. '07

I worked as a laboratory instructor teaching students about MS Excel, MS PowerPoint, and MS Access. During 2006-2007, I taught **System Analysis & Design (BUS ADM 436)** and **Information Technology Infrastructure for Business (BUS ADM 533)** independently.

MILWAUKEE SCHOOL OF ENGINEERING, MILWAUKEE, WI
Adjunct Assistant Professor, Electrical Engineering and Computer Science Dept.

March, '06 – Dec., '06

Taught advanced undergraduate course on **Computer Simulation and Modeling** and **Software Design II** which introduced students proficient in Java programming language to programming in C++ using object oriented concepts. I was also involved during the reaccreditation process by ABET.

SHEPHERD UNIVERSITY, Shepherdstown, WV
Part Time Lecturer, Department of Computer and Information Sciences

Aug. '02 – May '03

Courses taught - System Analysis & Design, and Introduction to Computer and Information Sciences.

INDUSTRY EXPERIENCE:

Various Companies

March '92 – August '03

I worked in the IS industry both as an employee for almost ten years and as an independent consultant for a little more than a year. I worked for various companies such as **Motorola, Tellabs, Cabletron, Putnam Investments** etc. I also worked in three different areas in the country – Tulsa, Ok; Greater Boston area and in Greater Washington area. I mostly worked in two industries – networking and financial. I worked as an individual developer as well as a team leader. During my tenure I participated in and observed the growth of the Internet. I have worked with C++ using object oriented technology. Some of the highlights of the projects I worked on:

- Developed software for implementing RSVP and MPLS protocols in C under VxWorks operating system for BSR 64000, an edge router with CMTS/POS/Ethernet interfaces and also tested interoperability with Cisco and Juniper routers, by configuring the routers. Also developed Cisco-like commands for configuring the router.
- Developed and tested device drivers for PCMCIA disk and Hard disk using C language for Everest, an ATM switch router. Also implemented a communication mechanism between different cards for the above router.
- Worked as the team leader for the mid-tier portion of the “Payroll Gateway” application of Putnam Investments. Payroll Gateway is based on three-tier client server architecture and it receives and processes transactions on incoming payrolls. Developed software using object oriented design in C++ for accessing the Oracle database through DBTools. Developed portions of data communication software in C++ between the desktop and the mid-tier using Orbix (A CORBA compliant software).
- Developed the User Interface for the diagnostics monitor and wrote device driver for EEPROM for an ATM switch, using C++ language and object oriented methodology.

- Developed C++ programming interface for use by applications to the ATM based network management system.
- Developed the GUI using ZINC interface library in C++ for a generic relay testing software “ULTRATEST” under DOS in Borland C++. Also developed the database and C++ interface to it, using PARADOX Engine.

TECHNICAL SKILLS:

- Data mining software – Weka.
- Strong Programming and Software Development skills using object oriented method in Java, C++.
- IDE – Eclipse
- ERP software - SAP
- Use of Network and Security tools – Wireshark, Nessus, GPG4Win.
- Mathematical and Statistical Software – Mathematica, SAS, SPSS, LISREL, Minitab.
- Use of virtual computing – VMWare.
- Power user of Course management Systems – Angel, Blackboard, D2L, WebCT.

PUBLICATION:

1. Jain, H and Dutt, Abhijit. 2013. “Economic Perspective of Application Development in Cloud Computing: Study of Modularity and Performance”. **International Symposium of Information Systems (ISIS)**, Goa, India.
2. Ngo-Ye, Thomas and Dutt, Abhijit. 2010. “Text Classification with Imperfect Hierarchical Structure Knowledge”. Proceedings of the **16th Americas Conference on Information Systems (AMCIS)**, Lima, Peru. (Presenting Author).
3. Ngo-Ye, Thomas and Dutt, Abhijit, "A Study on Efficacy of Ensemble Methods for Classification Learning" (2009). **ICIS 2009 Proceedings**. Paper 69. (Presenting Author).
4. Dutt, Abhijit. 2006. “Organizational adoption of Data Mining”. Proceedings of the **12th Americas Conference on Information Systems (AMCIS)**, Acapulco, Mexico. (Presenting Author).
5. Dutt, Abhijit and Nazareth, Derek. 2005. “Support for Wireless LAN Design”. **Proceedings of the 11th Americas Conference on Information Systems (AMCIS)**, Omaha, Nebraska. (Presenting Author).
6. Dutt, Abhijit and Srite, Mark. 2005. “A Cultural Perspective on Technology Acceptance”. **Proceedings of the 11th Americas Conference on Information Systems (AMCIS)**, Omaha, Nebraska. (Presenting Author).
7. Diaz, J. C. and Dutt, A. 1992. “Experiences with Line Ordered-Nested Block Preconditioning for Non symmetric Systems on the CM-2”, Proc. of International Conference on Computer Methods for Partial Differential Equations (IMACS – PDE 7).

PROFESSIONAL ACTIVITIES

- Associate Editor, ECIS 2012 and ECIS, 2013
- Member of the Campus Advisory Committee Penn State, 2012- Till Date.
- Member of the IST Advisory Board Penn State, 2008- Till Date.
- Member of the Academic Affairs Committee Penn State, 2010-11.
- Member of the hiring committee for instructor of IST, Penn State, 2010.
- Member of ASUG.
- Reviewer for the 30th International Conference on Information Systems (ICIS).
- Reviewer for the 11th and 12th Americas Conference on Information Systems (AMCIS).
- Reviewer for the Hawaii International Conference on System Sciences, 2004, 2005, 2006, 2007, 2009, 2010
- Member of the Ph.D. committee of the Lubar School of Business, UWM, 2005-2006.

AWARDS:

- Outstanding academic adviser award, Penn State Beaver, 2009-10
- UW-Milwaukee Graduate School Travel Award for attending AMCIS, 2006
- University of Wisconsin-Milwaukee Chancellor's Fellowship during 2003-2004. Was also awarded research and teaching assistantships along with tuition waiver during 2004-2007.
- Research assistantship along with full tuition waiver from University of Tulsa for pursuing MS in CS.
- National Scholarship from Govt. of India for pursuing both undergraduate and graduate studies.