CLOUD ADOPTION: A GOAL-ORIENTED REQUIREMENTS ENGINEERING APPROACH

by

SHEHNILA ZARDARI

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
14th March 2016

Abstract

The enormous potential of cloud computing for improved and cost-effective service has generated unprecedented interest in its adoption. However, a potential cloud user faces numerous risks regarding service requirements, cost implications of failure and uncertainty about cloud providers' ability to meet service level agreements. These risks hinder the adoption of cloud computing.

We motivate the need for a new requirements engineering methodology for systematically helping businesses and users to adopt cloud services and for mitigating risks in such transition. The methodology is grounded in goal-oriented approaches for requirements engineering. We argue that Goal-Oriented Requirements Engineering (GORE) is a promising paradigm to adopt for goals that are generic and flexible statements of users' requirements, which could be refined, elaborated, negotiated, mitigated for risks and analysed for economics considerations. The methodology can be used by small to large scale organisations to inform crucial decisions related to cloud adoption.

We propose a risk management framework based on the principle of GORE. In this approach, we liken risks to obstacles encountered while realising cloud user goals, therefore proposing cloud-specific obstacle resolution tactics for mitigating identified risks. The proposed framework shows benefits by providing a principled engineering approach to cloud adoption and empowering stakeholders with tactics for resolving risks when adopting the cloud.

We extend the work on GORE and obstacles for informing the adoption process. We argue that obstacles' prioritisation and their resolution is core to mitigating risks in the adoption process. We propose a novel systematic method for prioritising obstacles and their resolution tactics using Analytical Hierarchy Process (AHP). To assess the AHP choice of the resolution tactics we support the method by stability and sensitivity analysis.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Rami Bahsoon for his guidance, and encouragement. Thanks Rami for not only being my PhD adviser, but more importantly, for being an excellent support during my stay in UK. I would also like to thank my RSMG members, Prof. Peter Tino and Dr. Behzad Bordbar for their constructive and insightful comments.

Life in Computer Science has been a great pleasure due to the friends that I made during the course of my PhD. I really enjoyed technical and not-so-technical gossips with Sarah, Sakinah, Hana, Esra, Andrea, Catherine, Siti, Miriam and Bendra. I would specially like to thank my office-mates Momodou and Guanbo for being so patient (you know what I am talking about) and amazing.

My days at Birmingham would have been very dull, lonelier and torturous had Gurchetan Grewal not been my friend. I shared almost everything with him during this PhD and he always proved to be a great support. From crying on visa matters to sharing joy on paper acceptances, Guru had always been there for me. Guru, I will truly miss you. I wish you every success and happiness in your life.

I would also like to thank Dr. Rozmin and Dr. Munawwar Alam for being my second family in UK. You opened the door of your house and hearts for me. I cannot thank you enough for the love and care that you have shown me. Thanks to Dr. Rano Mal Hirani and family for hosting me in your house. I was in desperate need of accommodation and if you weren't there it would have been impossible for me to find a suitable place to live.

The acknowledgments would remain incomplete without thanking my father in law Mr. Maqsood Ahmed Shaikh (late). You truly believed in woman's emancipation. Gratitude to all my brothers and sisters for taking pride in me. That was indeed a driving force that kept me going.

Special thanks to my husband Navid Ahmed Shaikh for allowing me to stay away from him for so long and for taking care of Emaad while I was away in the UK. You have proved to be a really broad minded husband who believes in strong women. I wouldn't have come to the UK

for a PhD in the first place had you not encouraged me. Thank you so much.

Two special people who are very close to my heart and without whom I feel incomplete are Emaad and Shavez. You both are a blessing. Giving birth to two kids during PhD and raising them alone wasn't easy but you both are the most awesome boys. I hope this academic environment in which you both have been brought up leaves a lasting impression on your mind and that you both excel in your life. Nothing is as comforting as a cuddle and a moist kiss from you guys. You are my whole world. I love you both.

Finally, I would like to thank my parents who are not in this world anymore. It was their confidence in me which made me pursue my dream. I am fortunate to have had amazing parents. The time that we spent together will always be cherished. I would never have been a person who believed in herself had I not been trained in this way by my parents. I miss you both. Nothing can be as painful as not seeing your parents on your graduation day.

※※※※※

Publications arising from this Thesis

- S.Zardari, R.Bahsoon and Aniko Ekart, *Cloud Adoption: Prioritizing Obstacles and Obstacles Resolution Tactics Using AHP* in Proceedings of the 29th ACM Symposium On Applied Computing (SAC 2014), Gyeongju, Korea. *(Selection rate 24%)*

- S.Zardari, F. Faniyi and R.Bahsoon, *Using Obstacles for Systematically Modelling, Analysing and Mitigating Risks in Cloud Adoption* in the book on Aligning Enterprise, System and Software Architectures. IGI Global

- S.Zardari and R.Bahsoon, *Cloud Adoption: A Goal-Oriented Requirements Engineering Approach* in Proceedings of the ACM/IEEE International Workshop on Cloud Software Engineering, the ACM/IEEE 33rd International Conference on Software Engineering (ICSE 2011), Hawaii, USA

- S.Zardari, *Trouble in the cloud leaves businesses tied to their servers* in The Conversation, 15 May 2014.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

In my beginning is my end

East Coker, *T.S. Eliot*

Cloud computing is being heralded as the next big thing. Buyya et al. have defined cloud as:

"A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service level agreements established through negotiation between the service provider and consumer" [37].

The ever increasing need for data processing, storage, elastic and unbounded scale of computing infrastructure has provided great thrust for shifting the data and computing operations to the cloud. IBM advocates cloud computing as a cost effective model for service provision [82]. The adoption of cloud computing is gaining momentum because most of the services provided by the cloud are low cost and readily available. The pay-as-you-go structure of the cloud is particularly suited to Small and Medium Enterprises (SMEs) who have little or no resources for IT services [25]. The growing trend of cloud computing has led to many organisations and even individuals moving their computing operations, data, and/or commissioning their e-services to the cloud. Moving to the cloud has reduced the cost of computing and operations due to resource sharing, virtualization, lower maintenance cost, lower IT infrastructure cost, lower software cost, expertise utilization and sharing etc. [114]. For example, the New York Times managed to convert 4TB of scanned images containing 11 million articles into PDF files, which took 24 hours for conversion and used 100 Amazon EC2 Instances [71]. Such relatively quick conversion would be very expensive if done in-house. The term cloud computing may simply refer to different applications over the Internet or the hardware shared between different users [18].

In a cloud, hardware/software are shared and utilized as services at lower cost. The principle that underlies the cloud is provision of services on demand. These services could come with different flavors. Many services are now offered in the realm of cloud computing. Recent years have seen a tremendous increase in adoption of these cloud services. Despite the popularity of the cloud as a paradigm for providing services under the “utility” based model, little has been discussed in systematically engineering the adoption process. Regardless of the type of

cloud service, the fundamentals of adoption tend to be the same.

Adoption has covered several type of cloud services ranging from software, platform, infrastructure, data etc. as a service. For example in Software as a Service (SaaS), services are delivered as single application through the browser to thousands of users. Users are not required to invest in purchasing servers or software licensing. Payment is made on the basis of the data transferred and some fixed rent. Google App Engine is a representative example of SaaS. Another cloud model which is widely adopted is Infrastructure as a Service (IaaS). IaaS is a model in which an organization outsources the equipment required to perform operations like storage, hardware, servers etc. A cloud service provider provides all the hardware needed for operations and is responsible for maintaining it. The client pays for what he uses. Amazon's Elastic Compute cloud is an example of such a service. Among other models of cloud is Platform as a Service (PaaS) where a cloud provider provides a platform to the user on which a user can develop an application. The applications are delivered to the users through the cloud service provider's infrastructure. Coghead and Google App Engine are examples of Paas.

While cloud computing offers significant cost advantages, it can expose business organizations to significant risks. For example, cloud clients are often not given in-depth insights into the cloud provider's infrastructure, and supplier relationships. Such lack of oversight and control if not pro-actively evaluated and managed, already during cloud adoption, can lead to unexpected adverse costs to the business. Furthermore, Service Level Agreements (SLAs) are often one sided and non-negotiable, transferring all liabilities and cost to the client.

According to a 2013 Forbes survey, almost 69% of businesses have established separate cloud budgets, at a mean of 8.2M USD annually, and by 2016 the worldwide cloud computing market is expected to grow at an annual compound rate of 36% to a market size of $19.5B [3]. Although businesses are already embracing the cloud, it is unclear whether businesses, and in particular small to medium size businesses are aware of the range of risks the adoption of cloud computing can expose the business to. For example, according to the aforemen-

tioned Forbes survey, 30% of respondents cited security as their biggest concern during cloud adoption. However, the range of risks businesses can get exposed to is much broader, with security being an important “pain" point, but, certainly not the only significant one. Businesses currently seem to mitigate risks in a whole-sale manner by adopting internal clouds rather than taking more advantage of 3rd party cloud providers. Nevertheless, with a reported significant increase in projects for assessing adoption of external clouds as part of an overall IT strategy, there is clearly a need for offering a methodical approach to support the evaluation and management of cloud adoption risks to business organizations.

Client organizations typically have very limited access and knowledge of the cloud’s internal design. Infrastructure descriptions provided are often high-level and ambiguous. Another risk area is upgrades. Cloud providers retain full control over their systems’ upgrade choices and schedule. As a result, cloud users can be placed in unexpected adverse situations over which they have no control, leaving them at times with last resort choices only, of switching cloud providers. In cloud context, a service level agreement is expected to mediate consumers’ expectations with respect to cloud service provision. As a rule, the cloud is often a “black box", a user has little or no control over the promises set by cloud’s SLAs. Cloud users mostly are not given an option to negotiate the SLAs with the cloud provider and therefore must agree with the set terms and conditions. Despite the fact that businesses have already started to exploit the potentials of the cloud as a paradigm for services, there is an absence for foundation and methodologies for informing the process of cloud adoption to mitigate the risks.

1.1 Motivation

We looked at a case study of a leading cloud service provider referred to as *Indus* throughout this thesis. The case study revealed that there are many risks associated with cloud adoption. Since the cloud is perceived as a “black box", a user has little or no control over the promises set by cloud’s SLAs. For instance, the user cannot negotiate the SLAs with the cloud service

provider and hence has to agree with the set terms and conditions.

As an example, Indus sufficiently describes their security mechanisms in the agreement accompanying their SLAs “Indus Web Services Customer Agreement." Interestingly enough, Indus mentions that the nature of communication over the Internet is unpredictable and largely insecure. Given the vulnerability of the Internet, Indus cannot guarantee the security of users' content. While Indus strives for a secure environment, the security responsibility and accountability lie solely on the users and the organisation using the services. In the event of any breach in security requirements, Indus is not entirely liable to the user for any unauthorized access, use, deletion, corruption or destruction of the user's content. The service provider has attempted to win the confidence and trust of the users by publishing the agreement, yet it has failed to ascertain the individual needs of different users. Indus has a shared responsibility environment for the safety of user's content, where one of the inherent problems with such “joint" responsibility is that it makes accountability difficult. Referring to the SLAs terms and conditions, Indus absolves itself of any responsibility in the wake of anything going wrong. For instance, Indus recommends customers to encrypt data transferred over the network. There are tradeoffs involved with large data encrypted over the network: this will lead to higher processing time, which might affect cloud performance and consequently violate the promises set in the SLA. Despite the promised dynamic elasticity of cloud architecture, resources continue to be scarce. E.g. enhancing security provision may cause a performance bottleneck. As a result, the promised Quality of Service (QoS) often not be met as the SLA terms and conditions stipulate.

While service providers publish SLAs to win the confidence and trust of the users they fail to address individual needs of different users. Instead, broad solutions are suggested, at the expense of clients such as recommendations that customers encrypt data transferred over the network (e.g. section 4.2 of the Indus web services customer agreement). Careful reading of Indus Web Services' customer agreement shows that the company is not liable if your data is altered or deleted, nor if it is affected by any kind of security breach [10]. The only thing the

company promises in the service agreement is that the cloud will be available 99.95% of the time [11]. If it isn't, the customer gets compensation in the form of credit.

That means if the customer has paid upfront for using the cloud for ten hours, they are tied to the contract, even if the uptime is below 99.95%. All they get if the service was down is more time in the cloud. They don't even get a refund.

This is a significant problem that could put customers off. If a business has stored large amounts of data in a cloud service and there is a fault, the cost of the downtime - even if it is very short - may be significant. It might even be more than the investment made in using the cloud service in the first place.

Getting locked into the wrong cloud in this way is a major concern for users and prevents them from adopting the new service. Due to largely non-negotiable nature of service agreements, users have little recourse if things go wrong. The cloud service provider is always in a win-win situation. We identified a number of other risks that cloud users might come across. These included the potential for insiders with malicious intent to access their data; difficulty accessing the data for other reasons; and the need to comply with certain standards set by the industry. Problems like these could lead to financial losses, a loss of customer trust, damage to business reputation, losing company secrets and even the risk of going bust. The risks are huge and cannot be ignored.

We call for a novel requirements engineering methodology for cloud adoption, which could assist businesses in screening, selecting cloud service providers and negotiating their services and qualities of provision. The framework aims at helping businesses screen, match, and negotiate their requirements against cloud services' provision. The framework will also assist in the problem of managing the tradeoffs associated with matches and mismatches of users' requirements against cloud's provision. Such provision aims at objectively evaluating the strategic decisions, satisfaction of the technical and operational goals, cost and value of such decisions and the tradeoffs involved in moving to the cloud. Despite the rapid growth

of cloud use, there is a general lack of systematic methodologies aiming at screening the cloud according to user requirements. Decisions regarding the selection of cloud service providers are made on ad hoc basis based on recommendations or on the reputation of the service provider. The lack of such methodologies exposes businesses considering the cloud to unpredictable risks. It would be expensive to get “locked in" with a wrong cloud. Evaluating pre adoption choices at early stages is a cost-effective strategy to mitigate risks of probable losses due to wrong or unjustified selection decisions. Furthermore, the framework aims at assisting users in assessing their requirements against cloud provision. Due to the dynamic nature of the cloud, mismatches may occur between what is required by the user and what is provided by the cloud provider. The framework will assess the suitability of cloud service providers by exploring mismatches, managing risks, and suggesting possible tradeoffs. We use a goal-oriented approach for modelling user requirements for cloud provision. The expected beneficiaries of the work are small to large businesses, educational institutes and even individuals, who wish to exploit the cloud. Such work is novel and bridges an important gap in making the process of cloud adoption more transparent, systematic and user oriented.

1.2 Problem

The adoption of cloud services has gained momentum over the past few years. Almost everyone is signing up to the cloud due to its unbounded scalability and lower costs. Currently, cloud services are adopted after getting inputs from the existing users, cloud SLAs, cloud whitepapers etc. Service-based on demand applications change their features at runtime. Which means, that they have the ability to change thier functionality and quality of the service at runtime. Due to the frequent evolution of the cloud services it becomes extremely difficult for the cloud to adhere to user requirements. The inability of cloud services to satisfy user requirements may result in significant risks. The cloud providers’ SLAs are generally one-sided and static. Due to this static nature of the cloud SLAs, users usually have to compromise on

the set terms and conditions of the service level agreements.

We are the first to develop a framework which systematically helps users in screening and selecting the cloud according to user requirements [177]. Previously, users were unable to evaluate the cloud services and therefore had to sign up to the cloud, accepting the known and unknown risks that may arise due to failure of the cloud to meet their requirements. Since little is known about the internals of the cloud service provider, there has been increasing pressure for benchmarks to evaluate the cloud service at all times. Since cloud services are designed to address the market needs and not of one customer therefore a user will have to engage in requirements negotiation. Negotiation of requirements may result in new requirements and tradeoffs. The new set of requirements after negotiation may result in new risks, we therefore believe that requirements engineering and risk management for cloud is a continuous process. Earlier, there was a complete lack of a risk management framework with respect to user requirements in the process of cloud adoption.

1.2.1 Research Scope

To adopt a cloud according to user requirements it is necessary for the user to understand the SLAs provided by the cloud service provider and any known, and unknown risks that come with those SLAs. It is also important to understand that little is known about the cloud internals and therefore users have to evaluate the cloud according to the easily available information through SLAs, whitepapers, benchmarks etc. The evaluation process for selecting a suitable cloud involves the following group of stakeholders:

- *Cloud Service Consumer/User:* Any organization or user(s) who evaluate a cloud service according to the requirements.

- *Cloud Service Providers:* Companies responsible to provide cloud services of any kind such as Amazon Elastic Compute Cloud or Google App Engine.

The primary motivation of this thesis as discussed earlier is to investigate the process of cloud service evaluation and selection. Another important topic we aim to explore is the risk mitigation process during the cloud service evaluation and selection. More specifically, this thesis addresses the following issues:

Acquiring, refining and negotiating the requirements - Cloud computing is a very dynamic environment. User requirements change as rapidly as the features of the cloud services. It is therefore important to continuously involve the stakeholders in the requirements engineering process for the cloud. The generic requirements are first acquired from the users and later refined in the light of cloud SLAs. Since cloud services are not particularly designed for any user therefore, not all requirements may be met. The users should therefore be willing to negotiate their requirements against the cloud service provisions.

Risk analysis - The adoption of cloud services involves many risks. In order to make sensible decisions, users wishing to adopt the cloud services have to assess the risks involved while evaluating the cloud services.

Tradeoff analysis - When adopting cloud services, users have to make tradeoffs among requirements, cloud features, risks and costs. We aim to analyze and manage these tradeoffs. We would like to see how some tradeoffs may result into risks and how they can be mitigated.

Risk prioritization and mitigation - Not all risks may be critical. Some risks may be accepted due to their little impact on the users' system. This topic has been addressed with detail in this thesis. We have proposed some abstract and concrete risk mitigation tactics for cloud adoption. Risks have been prioritized using Analytical Hierarchy Process. To assess the prioritization of the risk and risk mitigation tactics we support the method by stability and sensitivity analysis.

1.3 Research Objectives

We briefly argued in the previous sections that numerous risks are involved when we adopt cloud services. An important question therefore is whether these risks can be mitigated or minimized through systematic engineering of requirements during the cloud adoption process.The objectives of this thesis are:

- To present a framework for acquiring, refining and negotiating user requirements while adopting cloud services.
- To mitigate the risks associated with cloud adoption.
- To propose a framework for screening and selecting cloud services according to user requirements.
- To propose techniques for mitigating the risks that may arise due to unfulfilled user requirements.
- It is important to rank risks according to their significance [29]. We therefore aim to propose a method for prioritizing and mitigating the identified risks.

1.4 Research Questions

We argue that evaluating the cloud services with respect to user requirements helps in mitigating the risks associated with its adoption. The thesis will therefore attempt to answer the following questions:

- Why a systematic requirements engineering framework is needed to help users in early informed decisions related to cloud service provider selection?

- How the framework would help users in identifying and mitigating the risks associated with cloud adoption?

- How will the proposed framework help in resolving the conflicts and managing tradeoffs among the user requirements?

- Will risk prioritization help in mitigating the risks effectively?

1.5 Contributions of This Thesis

The thesis is the first attempt to motivate a requirements engineering approach for cloud adoption. The claim is evidenced by citations and followup by other researchers [1]. In particular the thesis claims to be the first to formulate the adoption problem from the requirements engineering perspective. This thesis makes the following contributions

- The investigation has clarified the commonality and variability between classical requirements engineering (RE) and RE for emerging cloud paradigms. The thesis has questioned the relevance of classical RE definitions and their fit to the cloud. We have explored the shortcomings of the classical RE in the case of cloud. Based on these shortcomings we have motivated the need for a "customized" RE process for cloud adoption.

- This thesis builds on sound and influential work in RE to propose a systematic method which could help cloud users to make better informed decisions and uncover the potential risks in moving to the cloud. The method is grounded in goal-oriented requirements engineering. We have proposed a goal-oriented requirements engineering based method for modeling user requirements and using them for screening and selecting cloud service providers. The important phase of this method is matching of user

[1] Our paper,"Cloud Adoption: A Goal-Oriented Requirements Engineering Approach" has so far been cited 46 times.

requirements with the provisions of cloud service providers.

- Upon application of the method, the success of adoption depends on the extent to which the cloud provision meets user requirements. It is imperative that the matching is a complex process which can uncover many risks. The method is supported with a notation which can help in modeling the risks and systematically resolving them through appropriate tactics/risk resolution strategies which are grounded on GORE.

  Specifically, we have introduced the notion of obstacles in the cloud adoption process. Obstacles can be indicative of risks and should be resolved. We have proposed some abstract and concrete obstacle resolution tactics for mitigating the risks. We have also used utility theory for prioritizing obstacles and goals to determine their criticality in the system.

- The decision problem could become complex due to numerous requirements and the inability of the cloud to meet them. The mismatches between requirements and cloud features may be indicative of risks. Not all risks may be critical and therefore it is important to prioritize risks. After prioritization it can be decided whether risks are to be resolved or ignored. We have used AHP to prioritize obstacles and obstacle resolution tactics for risk mitigation. We have also done the stability analysis of the AHP choice of the resolution tactics.

- We evaluate the proposed framework on two non-trivial case studies to see the applicability of our method. We have found that our method has helped the organizations to model their requirements and have uncovered the risks that may have gone unnoticed otherwise. The proposed method has helped in prioritizing the risks according to their significance and has successfully resolved the risks through the proposed tactics. We have also evaluated different phases of the method under the criteria of subjectivity, scalability and completeness.

1.6 Structure of the Thesis

The overview of the thesis is outlined below

- **Chapter 2: Classical RE Vs RE for Cloud Adoption** Cloud computing has been an active research topic in software engineering in the past decade. Some publications have dealt with engineering requirements for cloud computing. However, they have not explored the differences between classical requirements engineering and RE for cloud computing. The goal of this chapter is to provide a comprehensive and structured overview of current requirements engineering practices and to identify the difference between the classical RE and cloud RE. Requirements engineering for cloud is a complex problem due to its ever changing features and volatile nature. The chapter motivates the need for cloud RE due to the differences between the traditional RE and cloud RE. The main objective of this chapter is to highlight the challenges faced by researchers and practitioners for adopting cloud services in the light of user requirements.

- **Chapter 3: Background and Related Work** The chapter gives background on GORE and current requirements engineering methods for cloud adoption.

- **Chapter 4: Cloud Adoption: A Goal-Oriented Requirements Engineering Approach**

  The chapter motivates the need for a new requirements engineering methodology for systematically helping businesses and users to adopt cloud services and for mitigating risks in such a transition. The methodology is grounded in goal-oriented approaches for requirements engineering. We describe the steps of the proposed process for screening and selecting the cloud services.

- **Chapter 5: Using Obstacles for Systematically Modeling, Analysing and Mitigating Risks in Cloud Adoption**

  This chapter motivates the need for a systematic approach to cloud adoption from

the risk perspective. The enormous potential of cloud computing for improved and cost-effective service delivery for commercial and academic purposes has generated unprecedented interest in its adoption. However, a potential cloud user faces numerous risks with respect to service requirements, cost implications of failure and uncertainty about cloud providers' ability to meet service level agreements. Hence, we consider two perspectives of a case study to identify risks associated with cloud adoption. We propose a risk management framework based on the principles of GORE. In our approach we liken risks to obstacles encountered while realising cloud user's goals, therefore we propose cloud-specific obstacle resolution tactics for mitigating identified risks. Our proposed framework shows benefits by providing a principled engineering approach to cloud adoption and empowering stakeholders with tactics for resolving risks when adopting the cloud.

- **Chapter 6: Cloud Adoption: Prioritizing Obstacles and Obstacle Resolution Tactics Using AHP**

  There are numerous risks which hinder the adoption of cloud. We extend the work on goal-oriented requirements engineering and obstacles for informing the adoption process. We argue that obstacle prioritization and their resolution is core to mitigating risks in the adoption process. We propose a novel systematic method for prioritizing obstacles and their resolution tactics using Analytical Hierarchy Process. We provide an example to demonstrate the applicability and effectiveness of the approach. To assess the AHP choice of the resolution tactics we support the method by stability and sensitivity analysis.

- **Chapter 7: Evaluation** Two different non-trivial case studies were used to illustrate the applicability of the proposed framework. We have evaluated each phase of the proposed method under the criteria of scalability, completeness and subjectivity. The proposed method has shown benefits by revealing the risks which otherwise could have gone unnoticed and through mitigating them.

- **Chapter 8: Conclusion and Future Work** we reflect on the journey of this thesis, and present concluding thoughts about directions that this research can take, in the future.

1.6.1 Publications

On page v, we provided a list of publications that were generated, during work on this thesis. This thesis contains, and must be considered, the definitive reference of details and ideas, present in those publications.

CHAPTER 2

Classical RE vs RE for Cloud Adoption

> Perhaps the central problem we face in all of computer science is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way. Science is supposed to be cumulative, not almost endless duplication of the same kind of things.
>
> *Richard Hamming*

2.1 Introduction

Adopting cloud and its services brings fundamental changes in how organizations think and engineer their requirements. These changes are induced by the characteristics of the cloud as a shared and dynamic environment motivated by the economies of scale. As a result, cloud and its services are designed to satisfy the generic and wider requirements of the market, instead of satisfying the requirements of a particular organization(s) or user. Despite the perceived benefits of the cloud, the environment introduces several challenges for requirements engineers. As an example, the generic nature of the cloud provision calls for an extensive process of evaluation, negotiation and prioritisation during the adoption process [179]. Unlike classical requirements engineering, these processes should cater for scale, decentralisation, uncertainties and heterogeneity making traditional approaches limited in their applicability. Consequently, interleaving the adoption process with requirements specifications tends to be a tiresome, complex task. Nevertheless, a successful adoption of the cloud with respect to user requirements depends on the effective and efficient evaluation, screening and selecting of all potential cloud candidates. The evaluation of cloud service providers involves matching between the user requirements and features of the cloud [177]. Mismatches or conflicts may occur between features of the cloud and what is required by users during the evaluation process of the cloud. Because of the conflicts which may arise during the evaluation of the cloud, a continual process of tradeoffs and negotiation should be performed. It is important that we balance the requirements of users with the features of cloud services. Any lack of adequate support for engineering requirements means that many screening and selection decisions tend to be biased based on the reputation of a cloud provider, their Service Level Agreement statements, past experiences and the like from subjective and ad-hoc inputs.

This chapter aims to explain the fundamental differences between traditional requirements engineering and requirements engineering for cloud. We first review some classical definitions for requirements engineering. We then discuss their limitation in addressing clouds' needs.

2.2 Motivation

Cloud computing allows the user to access data anywhere and anytime. In conventional computing the user can only access data from the system on which he has stored it. If the user needs to access the data in another system then he needs to store it on an external storage device. The user can, not only access his data anywhere but also work on a software without installing it on his system. Cloud computing therefore helps reduce the cost of software licensing and maintenance. User does not need to buy a new hardware for usage, instead he can use the cloud and pay for whatever space he is using for his data. Cloud computing allows users to pay for whatever capacity is needed unlike traditional computing which requires a lot of software and hardware establishment. So while traditional computing takes place on user's hard drive and on-site servers, cloud computing takes place on third party remote servers. Since there is a significant difference in a traditional system and a cloud based system, the requirements engineering for cloud is also significantly different.

We compare engineering requirements for cloud with classical RE. In traditional systems development, the requirements engineering activity involves obtaining stakeholders requirements, refining requirements into non-conflicting requirements statements and validating these requirements. The objective of a requirements engineer is to ensure that the requirements specification meets stakeholders' needs and represents a concise description of the system to be developed. The specified requirements are then used for software architecture design, implementation and testing. The requirements engineering process for cloud is different from those of traditional systems. One fundamental difference is the dynamic nature of cloud which also makes the RE process for cloud very dynamic. We present a detailed analysis of these differences. Requirements engineering for cloud can have two perspectives. The RE for cloud can be either for designing and deploying a service on cloud or it can be for adopting the cloud services. Our research is focused on engineering requirements for adopting cloud services.

Service-oriented computing and cloud computing are the reciprocals of each other. While service-oriented computing provides computing of the services, cloud computing provides services of the computing [166]. Requirements engineering for service-oriented architecture is different than for cloud adoption. Where Service-Oriented Architecture (SOA) is an architectural style for building a software. You can build an application by putting together different kind of services e.g. web services. The service-oriented architecture allows different kind of applications to exchange data as they participate in business processes [95]. Software as a service are basically stand-alone applications that offer business solution [95]. SaaS delivers software as utility services and charges on a per-use basis [95]. Service-oriented computing splits the developers into three independent but collaborative entities: the application builders (service requestors), the service brokers (or publishers), and the service developers (or providers) [150]. The responsibility of service developers is to develop software services that are loosely coupled. The service brokers publish the available services. The application builders find the available services through service brokers and use the services to develop new applications. The application development is done via discovery and composition rather than traditional design and coding [150]. RE for cloud is different than that of SOA. In the cloud, applications are owned by the cloud providers and the user can only execute it. In a SOA model the constituent components of the software are reusable services. In SOA, it's possible to combine services which results in creation of other services. Therefore, RE for cloud is heavily dependent on the matching of user requirements with the features of the cloud service provider. The services offered by the cloud are stand-alone which cannot be enhanced by consolidation of multiple services into a single composite service like SOA. Therefore, RE for cloud adoption looks at the features of the cloud for screening the cloud according to user requirements. Whereas, RE for SOA not just selects an existing service but can also build a new service by combining few additional services which are loosely coupled. According to [40] reusing existing components reduces the risk of introducing new failures into the process of enhancing or creating new business services. [111] identifies some of the risks across different viewpoints of an overall SOA-based Enterprise architecture. Risk

evaluation for SOA is a heuristic process [111] as the risks can be interdependent and can impact each other during implementation.

Cloud services are designed to cater to the needs of a mass market instead of satisfying the requirements of a particular user/organization and it is not guaranteed that available cloud will meet all requirements of the users. The major difference between classical requirements engineering and cloud RE is that the requirements for cloud do not need to be complete. Cloud and Commercial Off The Shelf (COTS) products exhibit similar characteristics, therefore the RE for cloud heavily borrows ideas from COTS RE like requirement flexibility, conflict management, negotiation etc. [7] [5] [6] [8]. RE for COTS has addressed the issues of conflicting requirements, tradeoffs, matching between user requirements and COTS features, and negotiation. Despite similarities between cloud and COTS there are significant differences that we have to consider while adopting cloud. In the case of COTS once the product is acquired it remains within the organization where the organization has full control over its functionality. Cloud can change functionality at will. According to Indus' webs services' customer agreement, SLAs and service terms may be updated from time to time [10] without any prior notice to the user. Cloud users have to rely on available benchmarks to see if the adopted cloud is functioning according to the promised SLAs. There is an increasing pressure for providing benchmarks for cloud services because unlike COTS products, cloud services which are offered as instances cannot be compared. As opposed to COTS where the organization knows how many features of a particular COTS product are being used, features of the cloud tend to be accessed on-demand instances.

To summarize, we can say that the objectives of this chapter are:

- To see how cloud services and traditional software are different. We would also like to see whether classical RE frameworks can be applied to the case of cloud or a different RE framework is needed.

- Although cloud and COTS exhibit similar characteristics, we need a different RE frame-

work for cloud; as cloud is not acquired as a product but is an instance based service.

- Due to dynamic nature of the cloud there are frequent changes in the cloud SLAs which effect the user requirements. We would like to know whether classical RE frameworks deal with this kind of service evolution and how this can be applied to the case of cloud.

2.3 Requirements Engineering for Cloud

Cloud services are instance-based offered under elastic and on-demand provision. It is often the norm that cloud providers are unable to guarantee similar performance for all instances due to fluctuating traffic and over-provisioned resources [83]. As a result, behavior tends to fluctuate and can't be guaranteed. Furthermore, though a cloud service may functionally match users' requirements, non-functional requirements are often difficult to achieve and maintain precision. Being a multi-tenant environment makes it extremely difficult if not impossible to cater for the wide variation of requirements; henceforth, challenging precision.

Cloud services are more complex due to their dynamic nature. Cloud services keep evolving at all times. Evolution is not a new phenomenon in software engineering. Classically some of the researchers and practitioners have used the term *"evolution"* as a substitute for maintenance. Evolution of software, open source software and COTS have been widely discussed by researchers. In open source software the system evolves when there is a change in the code by any volunteer programmer [70]. However, the originator of the software is free to decide which contribution(s) can be part of the final software [70]. Software evolution is related to the dynamic behavior of programming system as they are maintained and enhanced over their lifetime [24]. COTS products are susceptible to evolution, due to its need to achieve and maintain a broad customer base [50]. Even though cloud services and COTS both are designed to cater to the needs of a market instead of a single person or organization, there is a significant difference in how they evolve. COTS are acquired as a product whereas cloud

services are offered as instances. The instances are created on demand and may be different from each other. In a cloud environment, its not only the service which is dynamic in nature but also the number of stakeholders and their requirements. The number of stakeholders decrease/increase over time. Their requirements keep changing as they need to adjust themselves to ever changing cloud services. Evolution of services in the cloud case is therefore more complicated as compared to open source software and traditional software development as there is not one person who approves the changes made in the software. Due to the rapid evolution of cloud services and user requirements; requirements engineering for cloud is challenging.

Cloud services to a big extent are delivered as a “black-box" and hence the user has no control over the product. Due to the market competition and their dynamic nature, cloud services evolve frequently. Since little is known about the internal architecture, adopting a specific cloud type and provider may be considered risky. The user has to trust the cloud based on their available SLAs and market reviews (in some cases benchmarks). RE is particularly challenging for cloud adoption as the cloud is not adopted as a single product but instead an ecosystem of components (or technologies). This means that everything from the infrastructure to the software can be outsourced to meet the users' needs. Since cloud has a pay-as-you-go structure therefore it is only used when the user needs it and relinquishes the service when the job is done. Cloud services are offered in the form of instances. Instances are dynamically created: therefore their performance varies over different period of time. E.g the servers may be slow in responding during peak time. Cloud users have to rely on the available benchmarks to see if the adopted cloud is functioning according to the promised SLAs.

Cheng et al. in [41] have defined different research strategies in requirements engineering. One of which is *paradigm shift*. Paradigm shift in RE is something which dramatically changes the way of thinking which ensues a revolution in knowledge or technology. Requirements engineering for cloud is a paradigm shift keeping in view the necessary steps involved in RE for cloud. In the light of the classical requirements engineering definitions we compare the

cloud RE in Table 2.1

Table 2.1: Classical vs Cloud RE

Classical Requirements Engineering	**Requirements Engineering for Cloud**
Structured set of activities [138] which include requirements elicitation, requirements analysis, validation and documentation.	When adopting cloud for a very simple problem RE activities for cloud can be straight forward where all requirements can be satisfied by the cloud without negotiation. For a large and complex set of requirements the cloud adoption process may not be easy. There might be conflicts and tradeoffs between requirements which may require negotiation. Frequent evolution of cloud services results in frequent evolution of requirements [179]. RE activities are heavily dependent on the SLAs [179]. RE activities for cloud are highly iterative and cannot be structured due to highly volatile cloud services.
Influenced by social and organizational behavior. Discovering the purpose for which the system is intended and identifying stakeholders [124]	In case of cloud, users are scattered over different locations and therefore their social background and organizational behavior is difficult to gauge [92]. Instance based provision makes it difficult to obtain requirements. Stakeholders are distributed across multiple locations and its a daunting task to identify them.
Precise specification of software behavior and their evolution over time and across software families [180]	In cloud RE precise specification is difficult if not impossible. Since cloud is an instance based, open environment accessible to multitenant [113] users the behavior of the cloud cannot be precise and so its specification.
To provide a model of what is needed in a clear, consistent, precise and unambiguous statement of the problem to be solved [93]	RE for cloud has flexible requirements which can be negotiated over time against the cloud provisions [177]. Cloud requirements can never be precise. In fact, cloud SLAs may define new requirements after matching between user requirements and cloud services [179].

2.3.1 Requirements Flexibility

Cloud services aim to satisfy the generic requirements of the mass market instead of implementing the specific requirements of a particular customer. As a result customers may not get what is required by them and therefore should be willing to accept flexibility in their requirements. Thus, it is important to start with very generic requirements and then refine them in the light of the cloud features. This is to avoid excluding the potentially promising cloud services because of unnecessary constraints. Another approach is to let the available cloud service features/SLAs derive the elaboration of requirements [177]. This strategy recommends that the specification of requirements should occur together with the evaluation of the cloud services. The goals can be extracted from scenarios [15]. The requirements elicitation can begin with scenario elicitation where core goals can be identified.

2.3.2 Matching Between Cloud and User Requirements

The evaluation of the cloud services require some inexact matching between the user requirements and cloud service features [177]. Jan Bosch [30] has defined *features* with regard to software systems as: *"a logical unit of behavior that is specified by a set of functional and quality requirements"*. In the cloud case, features are the translation of what is promised by the cloud providers in their SLAs. There may be, for example, requirements not satisfied by any available cloud provider, requirements which cannot be satisfied by only one cloud and will be distributed to different clouds services (federated clouds). Requirements that are partially satisfied with features initially not requested but can be adopted after screening the cloud. There are situations where critical requirements cannot be fully satisfied without compromise on the requirements to accept cloud limitations. Requirements are left negotiable until the end of the selection process which allows the users to have a clear picture of which requirements can be implemented and therefore prevents them from having any high expectations from the cloud service provider. Keeping requirements flexible ensures that promising cloud providers

are not rejected in the beginning of the evaluation process just because their services do not satisfy some requirements. An important issue of requirements prioritization should be considered while specifying requirements for cloud adoption. One of the main objectives of the prioritization process is to differentiate between the desirable requirements that could be conveniently traded off when none of the available clouds meet all desired requirements from essential requirements that should not be compromised. Cloud service providers do make changes in their features in response to the market needs, however they are unlikely to change features to satisfy the requirements of individual users. For customers who are willing to make a large investment, the cloud service providers may be willing to make slight changes in their services/SLAs.

2.3.3 Negotiation for Cloud Adoption

Requirements Engineering for cloud does not involve building a system from scratch. Its about adopting the best available cloud in the market. As cloud services are not tailored according to the individual needs of a particular organization, much of the requirements engineering process for cloud is dependent on a matching and negotiation process between the cloud and user requirements. Negotiation can be either active or passive. In active negotiation the user negotiates with the cloud service provider or with an agent representing the cloud service provider [177]. Passive negotiation refers to when the user uses readily available information like benchmarks, expert judgment and reputation to assess the features of a cloud service provider [177]. The cloud service provider is prepared to enter into a negotiation stage if the value added of attracting an additional user is substantial or it outweighs the cost of negotiation. Figure 2.1 shows the negotiation process for cloud adoption. Requirements are negotiated against the provisions of the cloud service provider. Cloud services which cannot satisfy user requirements are discarded. User will finally adopt a cloud which is able to satisfy it's requirements after negotiation (see figure 2.1).

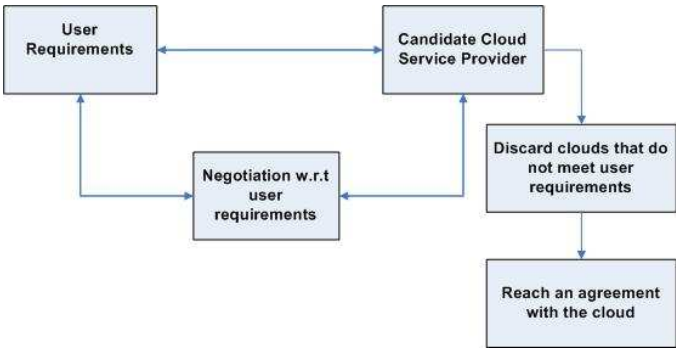

Figure 2.1: Negotiation for Cloud Adoption

2.3.4 Dilution of Control and Uncertainty

Cloud services are designed to meet the requirements of the widest number of possible customers. Thus, cloud services are designed to satisfy very general requirements. For customers to ensure that the candidate cloud satisfy at least their core requirements, they must have some knowledge about the cloud service features to decide which services must be adapted to satisfy their particular requirements. The fact that cloud services are largely delivered as black-boxes means customers have limited knowledge about the service internals and have no control over its operation. The information mainly available to the customer consists of market reviews and cloud service provider SLAs. As a result, customers have no guarantee whether the cloud service will perform according to the promises set in the SLAs.

A survey of different publicly available SLAs showed that while many cloud SLAs exist, there is little harmonization between different types, key elements and vocabulary [22]. This lack of universally agreed vocabulary among different cloud providers results in conflicting and biased information about cloud service and quality.

Very often cloud service providers make changes in their services, forcing customers to continuously upgrade their system. Such upgradation of the cloud features may increase the cloud fees as stated in the Amazon Web Service Customer Agreement (clause 5.1)[10]. Due to the new upgraded version of the cloud service conflicts may arise between different parts of the system. For example, certain security measures of the cloud user may conflict

with a cloud provider's environment, making their implementation by the user impossible [39]. Customers are therefore put into unexpected situations over which they have no control. Since the customer has no control on the release of new versions, they are left with no option but to upgrade the system or shift to another service provider. Changing the current cloud service for a new one can result in significant monetary loss. It is therefore easier for the customer to accept the cloud service provider's decisions and continue to upgrade the system.

2.4 Research Question

To understand the state-of-the-art in requirements engineering for cloud, in terms of gaps and commonalities in existing empirical results, we would like to know how major activities for the cloud selection can be mapped to a classical software engineering lifecycle. With focus on requirements engineering related activities, the objective is to position the requirements engineering related activity for cloud adoption in the overall context of development. As part of the exercise we have conceptualized how the commonly used software lifecycle such as waterfall and spiral development can be customized to the benefit of cloud adoption and its requirements engineering related activities. This chapter will therefore attempt to answer the following question:

- Why the existing requirements engineering methods are not suitable for the case of cloud and how cloud requirements engineering is different from classical RE?

2.5 Classical Requirements Engineering

Due to the characteristics that cloud imposes when engineering requirements, the requirements engineering for cloud is not as simple as the requirements engineering for traditional

systems. The traditional software system development starts with requirements engineering activity. [138] states that the term requirements engineering process refers to *"a structured set of activities which are followed to derive, validate and maintain a systems requirements document"*

The technical process of documenting and modelling requirements plays a vital role in requirements engineering. However, the requirements engineering process is also influenced by social and organizational behaviour [124]. According to [124] the software systems requirements engineering is *"the process of discovering purpose for which the system is intended by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation"*. This definition lays stress on the notion that requirements depict the desire of people affected by the system to be developed.

According to Zave [180] *"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families"*. This definition underlines the significance of goals as the driving force for the development of a new system. The definition is also meant to emphasize the role of RE in understanding the functional and non-functional requirements of the system (non-functional represented as constraints on the functionalities). The definition focuses on the specification activities taking both development and evolution across a family of products. According to Kotonya and Sommerville *"The main objective of the requirements engineering process is to provide a model of what is needed in a clear, consistent, precise and unambiguous statement of the problem to be solved"* [93]. According to this definition requirements should be clear and precise. Though RE aims at precision, consistency, clarity; these objectives are often difficult to fully satisfy due to unclear SLAs and frequently evolving cloud services.

Reflecting on Pamela Zave's [180] definition of RE, the definition is to a large extent concerned with the precise specifications of requirements, their realization as services and their evolution

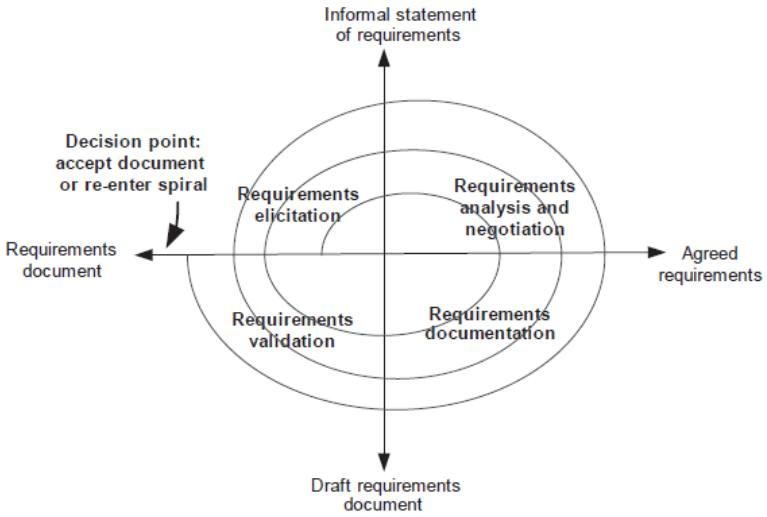

Figure 2.2: Spiral model of the requirements engineering process (adapted from[138])

over time. The definition also focuses on behavioral specifications and their precision. Despite the wide acceptability of this definition, it tends to be limited as when discussed in the context of cloud as precise specification is not idealistic.

[9] presented a spiral model which shows the main activities involved in the RE process. The model was adapted from [138] and is shown in Figure 2.2. The RE process starts with the elicitation of requirements ,which are then produced in the form of informal requirements. In the next step the requirements are analyzed and negotiated, once the requirements have been agreed the requirements are documented and stakeholders are asked to validate the document. The process is repeated until all the requirements are documented and validated by the stakeholders. Requirements keep changing during the system development and keep evolving once the system is in operation for some time [124]. The requirements engineering process varies from one organization to another. It is therefore important that this generic RE lifecycle is adapted to suit the needs of different organizations. The objective to review the classical RE definitions is to briefly describe the RE activities and their fitness to RE for cloud. We define each RE activity further in detail.

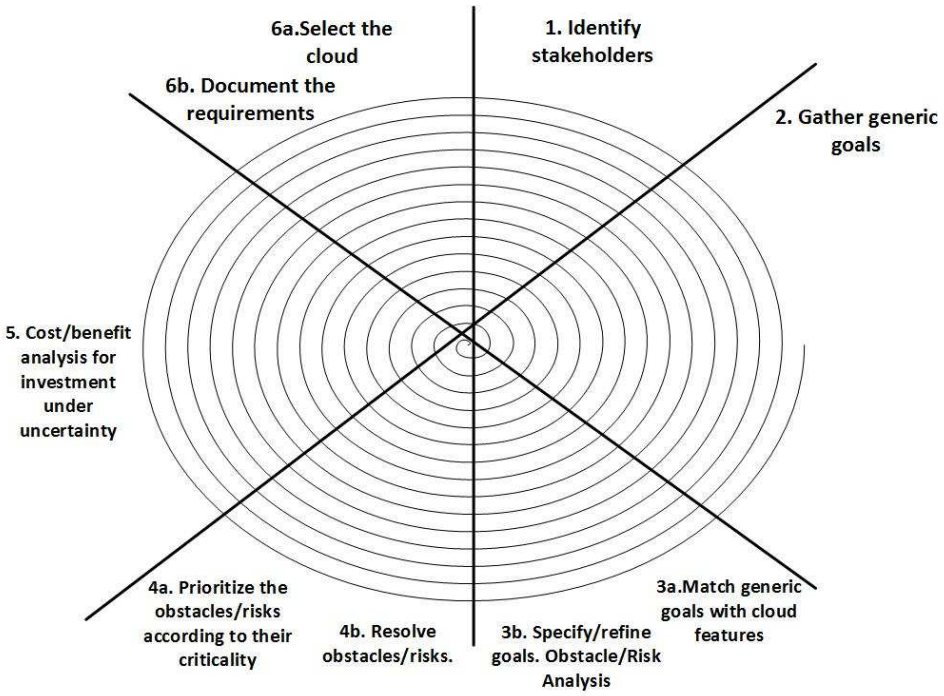

Figure 2.3: Spiral model for cloud adoption

Requirements Elicitation: The requirements elicitation activity deals with understanding of the goals and objectives for building a software system. The elicitation phase also involves identifying the goals which must be satisfied by the system. The activity also involves the identification of the concerned stakeholders and understanding their requirements and constraints. Stakeholders are individuals who are affected by the system [134]. Different stakeholders have different goals. Stakeholders may find it difficult to express their specific needs or they may set unrealistic goals. It is important that the requirements engineer elicit true requirements of stakeholders. The acceptability of the system depends on how effectively it meets stakeholders requirements [138]. Due to these reasons the requirements elicitation is a demanding activity. A number of requirements elicitation techniques have been proposed. [184] is a survey of different requirements elicitation methods available which includes elicitation techniques borrowed from ethnography, group techniques (e.g. brainstorming, focus groups) etc.

Requirements Analysis and Negotiation: Requirements analysis in software engineering includes those tasks that determine the needs or conditions to meet for a new or altered product, taking into consideration the potential conflicting requirements of the various stakeholders. These conflicting requirements have to be discussed and negotiated to reach an agreement between different stakeholders on how requirements can be changed. Conflicts should only be resolved once all the relevant information is available and it is possible to engage all the stakeholders in the negotiation process. The final set of requirements will be a compromise among the requirements of different stakeholders. Prioritization of conflicting requirements helps to facilitate the negotiation process of requirements. Requirements prioritization enables stakeholders to make tradeoffs acceptable by stakeholders among conflicting goals [169]

Requirements Validation: Requirements validation deals with careful check of the requirements to make sure that the requirements are complete and consistent [138]. The goal of validating the requirements is to check whether the requirements engineer had correctly understood the stakeholders' intentions and has not introduced any errors while writing the specification. If the stakeholders do not agree with the requirements specification then it is highly likely that they will not accept the final system. It is therefore important that all conflicts are resolved and stakeholders reach an agreement before the requirements are validated.

Requirements Management: Requirements management is the process of managing the requirements that change through the system lifecycle. According to [124] managing requirements does not only mean managing the requirements documentation but it means managing the requirements documentation in such a way that it is readable and traceable by many, so that requirements evolution can be managed over time. Mostly changes in require-

ments occur due to the following reasons: errors and inconsistencies in requirements [63]. A major factor to help the management of requirements is the ability to trace the requirements [124].

2.6 Requirements Engineering for Cloud Adoption

Cloud adoption brings several new challenges and risks to organizations. These challenges and risks will, in turn, affect the cloud adoption process. The market-driven nature of cloud services brings circumstances that did not exist in the traditional in-house system development. In the classical software development, the boundary between development phases is clearly defined. Software development starts with the specification of requirements, then the system architecture and design are defined and the software system is implemented. Even if spiral-type process models are adopted [28], it is quite easier to differentiate the limits between each phase. On the other hand, the cloud adoption process consists of highly interactive and iterative processes, where the distinctions between the activities of requirements specification, architecture design and cloud evaluation/integration become blurred. Spiral model due to its continuous evolution of requirements and risk management can be adapted to the case of cloud. Agile software development also welcomes changing requirements late in the development [78]. The software product are delivered frequently within as much shorter time as possible [23]. Although agile supports requirements evolution however, it strongly supports interaction between the stakeholders [23]. In the cloud case, where the stakeholders are scattered and have a diverse background, such face-to-face interaction is not possible. While part of the agile principles are suited to the cloud case, agile cannot solve the issues related to heterogeneity of cloud users. Resolving conflicts among user requirements, negotiation between the cloud service provider and user are challenging as users are working at different locations and cloud service providers usually have one sided non-negotiable SLAs.

Figure 2.4 suggests that the traditional development process uses the waterfall model [137].

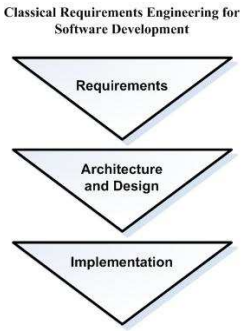

Figure 2.4: Classical RE

However, we consider that even in traditional software development, spiral models [28] are more widely used and appropriate than the obsolete waterfall model. Despite the differences between the traditional and cloud based requirements engineering, the RE for cloud adoption involves a number of activities that are also present in the traditional RE for software development. However, these activities may suffer changes in the context of cloud. As we discussed earlier, the flexibility in requirements imposed by the use of cloud services brings completely new situations to the requirements process. Although some traditional requirements techniques are still effective, new approaches are needed to suit specific circumstances of requirements engineering for cloud. The frequent modifications in cloud services affect the system architecture and design in such a way that the architecture has to be sufficiently flexible to accommodate evolving requirements. The implementation activity in the classical RE method is replaced by the integration/adoption of cloud services. The system maintenance after cloud adoption includes timely upgrades and cloud provider switch if needed. It is important to continuously perform requirements analysis and do risk management as the requirements engineering for cloud is not a one time affair. RE for cloud is a continuous activity because of the dynamic nature of the cloud and ever changing user requirements. RE for cloud adoption can therefore be defined as:

"Requirements engineering for cloud is a continuous process of gathering flexible core requirements which can be refined and negotiated against the cloud service provider's features. Cloud RE must also take into account uncertainty, possible risks and risk mitigation strategies which may result in new requirements."

2.7 Accommodating Changes

Developers must be able to deal with a new set of problems and situations. In the traditional development paradigm, developers had a reasonably dominant role over the development of the new system. Their major tasks involved producing the requirements specification, defining architecture and design models, implementing the software system according to the requirements and maintaining the system to accommodate future changes.

The fundamental difference in the role of a developer in cloud adoption is that he does not need to focus on the implementation of the system, rather he needs to concentrate on evaluating the cloud services with respect to user requirements. Different users have different demands and hence different results after evaluating the same cloud. Cloud services are generic and therefore cloud may not be able to satisfy all user requirements, it is for this reason that users should be prepared to engage in extensive prioritization and negotiation of their requirements. It is important that users are actively involved in the cloud adoption process and contribute to the decision making process.

During the evaluation of the cloud services the evaluators must have the skills to measure the technical competence of cloud services (which can be gauged from benchmarks,cloud white papers, SLAs etc.) as well as have the competence to assess more subjective issues like the reputation of the cloud service provider. Moreover, evaluators have to deal with issues such as making tradeoffs between available cloud services and requirements.

2.8 Risk Analysis of the Cloud

There has been little work in the area of risk management for cloud. [182] presents an information risk management framework for better understanding the critical areas that must be focused in cloud computing environment, for identifying a threat and vulnerability. The

framework covers all of cloud service models and cloud deployment models. This framework however is generic which does not take into account risks arising from unfulfilled requirements of a particular customer.

Cloud consumers often have to abide by the laws that prohibit users from sharing their data outside a certain territory. Therefore, if a consumer does not know where the data is being stored and processed may have serious implications. In this context risk management and SLAs have become very important issues surrounding cloud computing. [116] has attempted to identify the issues and their corresponding challenges. [116] has proposed risk and SLA management to improve governance, compliance and risks in cloud computing environments. The identified risks are mainly related to general security and compliance. We beieve that conflicting requirements, tradeoffs and unfulfilled requirements may be indicative of risks. We therefore analyze the individual requirements of every user to probe for risks.

[64] has presented a risk-management procedure specific to the cloud to determine risks impact on Business Level Objectives (BLO). The authors have proposed a method which has a main goal of prioritizing the risks according to their impact on different BLOs. The authors have observed that cloud service providers can maximize their profit by transferring private cloud's provision risks to third party cloud infrastructure providers. Again, the risks are generic and have not been identified from the user requirements.

[32] contributes to an empirical study of a sample of Swiss companies which is aimed at analyzing the understanding of the risks that public cloud services present and how they can be managed. In [32] the authors elaborate on the most important risks inherent to the cloud such as information security, regulatory compliance, disaster recovery and data location.

We argue that risk management for cloud cannot be generic. We first need to analyze individual risks and then mitigate them. There is no one solution to all the identified risks. We therefore believe that risk management should be the part of the RE phase for cloud adoption. Some of the unfulfilled requirements may be indicative of risks. We believe that risks should

be mitigated at the RE level so that they don't pass on to the final system.

We proposed a risk management framework based on the principle of GORE [179]. In this approach, we liken risks to obstacles encountered while realizing cloud user goals, therefore proposing cloud-specific obstacle resolution tactics for mitigating identified risks. The proposed framework showed benefits by providing a principled engineering approach to cloud adoption and empowering stakeholders with tactics for resolving risks when adopting the cloud. We prioritized the risks and risks mitigation tactics through a novel systematic method using Analytical Hierarchy Process [178].

2.9 Conclusion

The methodology that was followed for deriving that comparison is grounded on literature review and evidence on the distinction as gathered from several cloud and non-cloud related papers. As cloud is becoming increasingly popular this research is timely. Our research is answer to the question that why existing requirements engineering method are not suitable for the case of cloud and how cloud requirements engineering is different from classical RE. Requirements engineering for cloud does borrow the ideas from classical requirements engineering however, it is much more challenging due to the dynamic nature and unbounded scale of the cloud services. Unlike the classical requirements engineering the RE for cloud cannot be complete. Requirements are negotiated against the cloud SLAs. Cloud SLAs keep changing over time and so do the user requirements. Even if we use the classical methods of requirements elicitation like scenarios, brain storming sessions and interviews we would only be able to get partial set of user requirements. It is due to this reason that requirements for cloud need to be flexible which can be negotiated against the available cloud features.

We have discussed that the requirements engineering for cloud adoption differs from the traditional requirements engineering for software development. One of the activities exclusive

to that of cloud adoption is the selection of cloud service. This is considered to be a core activity of the cloud based requirements engineering. Matching between cloud features and user requirements may be indicative of some potential risks which arise due to unmet user requirements. Thus, risk management should be part of the requirements engineering process for cloud adoption. Possible flavours but not limited to, that can fit the RE for cloud process are spiral and waterfall models. When the requirements are not complex and no iterations are involved users can opt for waterfall model. The more complex the features and requirements become we move towards spiral model. We further explore in Chapter 3 the current requirements engineering methods being used for cloud adoption.

CHAPTER 3

Background and Related Work

3.1 Introduction

Requirements engineering for cloud is more challenging as compared to engineering requirements for a traditional system. It's due to the dynamic environment of the cloud where continuous evolution of services is the norm. User requirements also keep changing due to the ever changing cloud services and their SLAs. Requirements are basically the way of achieving a goal; it is therefore possible that requirements may evolve towards another way of achieving the same goal. Hence, goals are more stable than requirements [14]. A goal tree also provides traceability links from high-level goals to the low-level technical requirements. This traceability allows users to identify any missing requirements and risks. GORE helps in identifying and managing any conflicting requirements [160]. Due to heterogeneity of users the requirements for adopting cloud may have conflicts. GORE will therefore help in resolving

the conflicts. Requirements for cloud cannot be complete as little is known about the cloud internals. However, GORE provides a criterion for requirements completeness by achieving all goals [172].

Cloud services aim to satisfy generic requirements of the mass market instead of implementing specific requirements of a particular user. Therefore, the RE for cloud heavily depends on matching between user requirements and the cloud service providers' features [177]. Any mismatches between user requirements and cloud features may be indicative of risks [177]. There may also be some obstacles in the achievement of user goals. The obstacles if not resolved may turn into risks [179]. GORE helps in identifying and resolving those obstacles [163].

It is important that requirements will keep changing in cloud context. As a result, GORE can provide better support for evolution where we can concentrate on analyzing part of the tree which has been directly or indirectly affected by the changes in SLAs or user requirements. Thus, the analysis could be reiterated to discuss the service ability to meet the changes in the requirements. We argue that GORE is a promising paradigm to adopt for goals that are generic and flexible statements of users' requirements, which could be refined, elaborated, negotiated and mitigated for risks [177].

3.2 Use Case-Based Requirements Engineering

Use cases describe possible interactions between a system and its environment [136]. According to Unified Modeling Language (UML) specification, a use case is " the specification of a sequence of actions, including variants, that a system (or a subsystem) can perform, interacting with actors of the system" [140]. A use case describes a piece of behaviour of a system without revealing the system's internal structure. Practitioners and scientists have proposed different methods for eliciting and analyzing software requirements [135]. A use

case model includes use cases, actors and relationships. A use case diagram is a graphical depiction of a use case model in the UML. Use case diagrams show use case names, actors, relationships between actors and use cases, and relationships between use cases. Use cases are mainly being used for modeling functional requirements [142] using UML. Use case modeling models the relationships between different actors and requirements. However, the use-case does not deal with the non-functional requirements. For screening the cloud non-functional requirements are as important as the functional requirements. For example, the organization trying to adopt cloud may have some budgetary constraints. Use cases also don't explicitly model the risks. We therefore use goal-oriented methodology for acquiring and specifying user requirements. The requirements captured can be either functional or non-functional. GORE also provides a systematic method for identifying, analyzing and mitigating the risks in the form of obstacles. Use case tries to capture the behaviour of a system with actors. Since cloud is a volatile environment, the actors involved in the RE process will keep changing. Thus, the use cases are very difficult to be modelled for cloud. In a cloud case, the behaviour of every instance created is different and cannot be predicted. Cloud needs an RE framework which can screen and select the cloud according to user requirements while also mitigating the risks.

3.3 Feature Modeling

Software product line (SPL) engineering is a paradigm for systematic reuse [84]. From common assets, different programs of a domain can be assembled. Programs of an SPL are distinguished by features, which are domain abstractions relevant to stakeholders and are typically increments in program functionality. Every program of an SPL is represented by a unique combination of features, and an SPL could have millions of distinct programs. A feature model compactly defines all features in an SPL and their valid combinations; it is basically an AND-OR graph with constraints. The selection of the most suitable features for a

specific application requires the understanding of its stakeholders' intentions and also the relationship between their intentions and the available software features. To address this issue [19] has adopted a standard GORE framework, i.e. the *i** framework, for identifying the stakeholders and has proposed an approach for mapping and bridging features of a product line to stakeholders' goals and objectives. This is similar to our approach where we match the user requirements acquired in the form of goals with the features of the cloud service provider. However, feature models themselves don't deal with the risks. Our proposed method is helpful in probing the risks resulting from the mismatches between user goals and cloud features. Also, feature models help in making a software product where different programs can be assembled together. In a cloud case, we adopt a service without being able to assemble it to another service. Cloud services are stand-alone which cannot be enhanced by consolidation of multiple services. The RE for cloud therefore involves matching between cloud features and user goals, requirements and SLA negotiation, risk identification and mitigation. Also, stakeholder identification is not so straightforward in the case of cloud. Since stakeholders scattered and have a diverse background, there will be conflicting requirements. Our proposed method also deals with the conflicting requirements in the cloud adoption process.

3.4 Goal Oriented Requirements Engineering

The influence of goal-oriented requirements engineering is apparent on existing RE methods and techniques [152]. A large number of RE approaches have used the notion of goal as high level abstraction for structuring the requirements [13] [48] [127].

A comprehensive review of goal-oriented requirements engineering can be found in [153]. Requirements are represented in the form of goals in GORE. GORE approaches have normally the following activities: goal elicitation, goal refinement and various type of goal analysis, and finally the assignment of goals to the agents such as humans, devices and software. Requirements engineering research has increasingly accepted the pivotal role played by goals

in the RE process[128][180][118]. The recognition of goals in RE has led to a whole stream of research on goal modeling, goal specification and goal-based reasoning for multiple purposes, such as conflict management, tradeoff analysis [8], risk modeling [15].

Goal-oriented requirements modelling is an important approach for specifying the requirements. Goals have been accepted as a key concept in the requirements engineering process [153] and this area has gained popularity over the last few years. A goal is an objective which a system under consideration should achieve through cooperation of agents.

3.4.1 What are Goals?

There has been several definitions of goals in RE literature. Long before goals were introduced in RE, they were used in artificial intelligence (e.g., [122]). Yue [176] was the first one who argued that goals modeled explicitly in requirements model provide a criteria for requirements completeness. A number of definitions have been proposed for goals in current RE literature. Goals have been defined differently in different context. Goal definition has been adapted to suit different objectives.

Different types of concerns are covered by goals [153]. Functional concerns related to the services to be provided and non-functional concerns related to the quality of service like performance, security, safety etc. Goals have several definitions in classical RE. Among the most cited definitions of goals in the requirements engineering literature are the ones by van Lamsweerde [153] and Pamela Zave [181]. van Lamsweerde describes goals as,
"A goal is an objective a system must achieve".
Customers will have to engage in negotiation and will have to let go of some of the requirements. Goals may always not be only of *achieve* type. Customers may have the goals where a certain level of quality is to be maintained, some action is to be avoided etc. The classification of goals is further defined in Section 3.6.6.1. Zave et al. have defined goal as,
"Goal formulation refers to intended properties to be achieved, they are optative statements as

opposed to indicative ones, and bounded by the subject matter" [181].

According to Zave goals are optative statements which are defined in the context of a domain. In the case of cloud, where the system is open ended and highly volatile, it is difficult to define the goals with respect to the domain. According to Antón [16],

"Goals are high level business objectives of any organization or a system. They provide a rationale why a system is needed and guide decisions at every level within the organization."

Anton's definition says that goal decomposition is a top-down approach. Top-down decomposition of the goals require well formed questions and definitive answers (operationalization alternatives)[87]. In cloud case, goals are iteratively defined and refined, keeping in view the features of ever changing cloud services. Therefore, instead of being top-down, the goal operationalization is highly iterative. Dardenne et al. [48] define goals as:

" a nonoperational objective to be achieved by the composite system"

Dardenne's definition says that goals are high-level and will be refined to be operationalized by the system. Literature however shows that goal-oriented requirements engineering has been used to model the functional requirements associated with the services that are to be provided [153].

The literature has defined goals in the context of agents for carrying out the job. The definitions for the goals have been approached from two perspectives, design and runtime. Example of runtime includes self adaptive systems. [34] has used KAOS approach for modeling adaptive semantics in adaptive systems. Adaptive semantics are meant to define how a system behaves during adaptation. [34] defines a goal as:

"A requirement is a goal under the responsibility of an automated component"

This definition is limited to only functional goals which can be operationalized after assigning them to an agent. But as we have discussed earlier, goals do not necessarily need to be assigned to an agent. Goals can be of various types e.g. business goals/strategic goals, functional goals, non functional goals etc.

Goals change due to uncertainty about the future environments in which the system may operate. [35] has presented an approach for assessing the stability of requirements in long-lifetime systems using goal-oriented approaches. Scenarios are generated for any possible future environments and stability of the requirements is assessed in those scenarios. This approach however may not be helpful in the case of cloud due to the highly volatile and unpredictable nature of the cloud services. Scenarios for future changes in the cloud cannot be predicted beforehand because of the little knowledge customers have about the cloud internals.

Requirements engineering for cloud is therefore incomplete due to the black box nature of the cloud and goals cannot be stable due to the ever changing cloud features. User requirements for adopting cloud services can be obtained in the form of goals [177]. At first high-level goals are elicited and then they are refined into more concrete goals. The goals are kept generic in the beginning so that during matching of the cloud services with the goals, promising cloud services are not left out due to very precise requirements.

3.4.2 Goal Elicitation

Goal identification is not an easy task [161] [17] [77] [129]. Goal elicitation is the process of seeking, acquiring, and modeling user requirements. Goals can be elicited during the forward engineering phase and the reverse engineering phase.

For more than a decade the requirements engineering community has been studying goal models [48] [46] [118] [155] as high level abstractions for modeling initial requirements. Usually it is argued that goal models are built during the initial phases of the RE process [173] [48] [54]. Requirements “implement" goals like the program implements the design specifications [161] [144]. The classical definitions suggest that goals can be elicited in the initial phases of system development. Goal elicitation can support the forward engineering of a system under development, where the elicitation process can help in identifying requirements that need to

be realized by the design of the system.

Usually, stakeholders have some goals in their mind about the system-to-be[12]. Goals may be explicitly defined by the stakeholders through different material to the requirements engineer. Often goals are implicit and therefore goal elicitation must take place. An important source of goal elicitation is the preliminary analysis of the existing system. This analysis of the existing system usually results in a list of deficiencies and problems which can be precisely formulated. [157] suggests that by negating these formulations we can get a first set of goals that need to be achieved by the system.

Literature on goal-oriented suggests that interviews, scenarios, policy statements, use cases etc. can be the main source of goal elicitation [14] [161]. In [13] it is suggested that stakeholders tend to specify their requirements in terms of actions, instead of goals. So, the requirements engineer should look for action words such as “apply", “interview", or “reject" for a job application system. Section 3.4.2.1 explains how goals can be elicited from scenarios.

Elicitation can also take place to support reverse engineering activities, where goals can be interpreted from exiting legacy systems and mapped to the system to be. Preliminary analysis of the existing system, in the case of reverse engineering, would focus on analysis of the operation of the legacy system along with requirements of the system to be. Goals can be elicited for the sake of reverse engineering or system maintenance. [175] has proposed an approach to produce goal models from the source code of legacy systems. Goal models are produced through code refactoring and immediate state-based models. Goals are elicited during the reverse engineering phase. [103] has proposed taking a customizable personal software system and creating a goal model systematically which represents a refinement of high-level user goals into the configuration options that exist in the software.

3.4.2.1 Goals and Scenarios

[165] defines scenarios as a temporal sequence of interactions among different agents in the restricted context of achieving some implicit purpose. Scenarios have been used in requirements engineering (e.g. [143] [126] [13]) and have proven to be useful for goal elicitation and obstacle analysis. Scenarios are easily comprehensible by the stakeholders because they are informal, narrative and concrete descriptions of hypothetical interactions between the software and its environment. A number of goal-based approaches have embraced scenarios. Scenarios can be used for goal elicitation [165]. Scenarios have also been used to identify the obstacles which could hinder the satisfaction of a goal [13] [126].

A number of approaches have used scenarios together with goals. In [126], the author has derived the scenarios from the description of the system's and users' goals, and the potential obstacles that might hinder the achievement of those goals. Potts [126] has proposed a scenario schema and a method for obtaining a set of salient scenarios, scenarios that have a point and help people understand the system in question. The reason behind scenario salience is to limit the number of generated scenarios.

In KAOS context, Lamsweerde and Willemet [165] proposed a formal approach to infer specifications of system goals and requirements inductively from interaction scenarios. The method takes the represented scenarios as UML sequence diagrams as examples/counterexamples of the system-to-be behaviour, and inductively infers a set of goals that includes all example scenarios and excludes all counterexample scenarios. The aim of goal-scenario integration in KAOS is to get additional goal specifications from scenarios that otherwise could not be obtained from the goal refinement process that began after initial goal identification from interviews or existing documentation. Eliciting goals from scenarios however may not be very helpful in the case of cloud as the features of cloud are not completely known to the users and also because of the frequent evolution of the cloud services which make it very difficult to predict the future scenarios.

According to Anton [14] goal analysis starts with the identification of goals from the scenarios. [14] provides heuristics for identifying goals from scenarios. Examples of heuristics for identifying goals from scenarios can be found in Table 3.1 We shall not go into much detail of the scenario analysis as this is outside the scope of this thesis.

Table 3.1: Heuristics for identifying goals from scenarios [14]

No.	**Heuristic**
H1	Look for action words like satisfy, increase, achieve, avoid, amend to identify the goals?
H2	Stakeholders usually express their requirements in the form of actions instead of goals. It is therefore beneficial to look for action words to extract goals from stakeholders' descriptions.
H3	Customers define their goals in the context of their application domain, not in the context of a desired or existing system. Requirements engineer should therefore first understand the application domain and goals before focusing on the desired system so that the actual set of requirements can be elicited.
H4	It is important to uncover hidden goals by considering each action word and asking "why" until all the goals have been addressed. This process will make the analyst confident that the rationale behind all the actions have been understood and expressed as goals.

3.4.3 Goal Refinement

Goals can be refined by asking HOW and WHY questions about the already available goals [161][152]. Higher level goals are elicited by asking WHY questions about the available goals. After the initial set of goals have been obtained, the aim is to refine them into simpler goals until the subgoals can be operationalized. Operationalization of goals is possible by assigning them to individual agents [153]. This process is done by asking HOW questions. The goal refinement tree shows the relationship between subgoals and the parent goal using AND/OR refinement. AND refinement defines that a goal is only satisfied when all its subgoals are satisfied. OR refinement means that a goal is sufficiently satisfied if any one of the subgoals is satisfied.

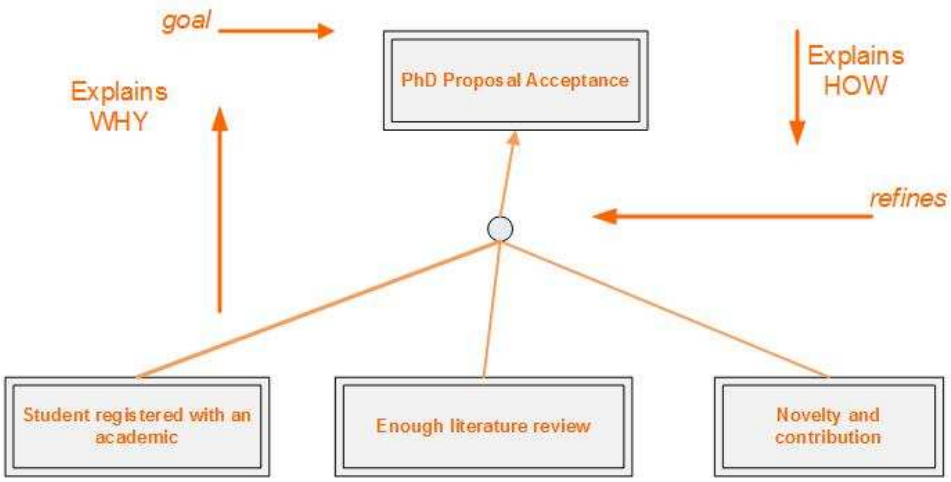

Figure 3.1: Example of goal refinement

As an example, figure 3.1 shows the refinement of the goal for PhD proposal acceptance into subgoals. To achieve the main goal, subgoals must be satisfied. Student needs to be registered with an academic for a PhD proposal acceptance and should have done enough literature review along with having novelty in his proposed work.

3.4.4 Goal Operationalization and Satisfaction

Functional goals allocated to software agents are required to be operationalized into service specifications that agents need to provide in order to achieve them [118][48]. Non-functional goals about quality of service are used to choose goal assignment alternative that satisfies them the best [46]. Operationalization is a process of mapping declarative property specifications to operational specifications that are able to satisfy them [99]. Different goals are operationalized differently according to their classification. Any goal which is related to security and privacy are required to avoid a certain event from occurring through some actions. On the other hand, if services are demanded at any time the goal is thought to be operationalized if the required services have been provided [49]. In the cloud case, goals are operationalized after they have been assigned to the agents (cloud service) which are able to satisfy user requirements.

[101] proposed a quantitative approach for reasoning about partial goal satisfaction. Goals do not necessarily need to be fully satisfied. A statement “the cloud should at least be available to users for 95% of the time" is an example of this phenomenon. [9] has defined the concept of acceptable range to determine the values in which operational goals are considered to be sufficiently satisfied.

3.4.5 Assigning Goals to Agents

Goal refinement stops when all subgoals can be allocated to agents [99]. Agents are components such as humans, devices, legacy software or software-to-be components that play a role for goal satisfaction [101]. Some goals may therefore define the software whereas others define the environment. In our case the agents can be either cloud service providers or a particular service. Goals can only be assigned to the agents which are able to satisfy them. [98] has proposed a taxonomy of realizability problems and from this taxonomy the authors have derived a catalog of tactics for resolving these problems. Though [98] provides support for generating alternative assignments and refinements through alternative tactics; it provides no support for evaluating alternatives and selecting the the best one.

3.5 Why GORE?

There are number of advantages associated with goal modeling, goal refinement and goal analysis [153].

- Conflict management of *viewpoints* is one of the concerns of RE [56]. Goals can provide the basis for identification and management of conflicts between requirements [160] [151]. [163] has studied in detail a particular type of conflict called divergence. The approach is based on management of inconsistencies at the goal level so that conflicts

in requirements are not further propagated to the system design.

- Goals provide a perfect way for communicating requirements to the users. Goal refinement offers the right level of abstraction for involving decision makers to validate choices to be made between different alternatives and for proposing other alternatives [152].

- A goal tree gives traceability links from high-level objectives to low-level technical requirements which can be assigned to agents [100] [172]. Particularly for business applications, goals may be used to relate with the software-to-be to organizational context [174].

- GORE addresses the early requirements engineering phase during which stakeholders' goals are acquired and different options available for achieving these goals are explored [96] [100].

- Finding the alternative system solutions is at the heart of GORE approaches to RE [96].

- GORE also deal with non-functional requirements (such as performance, security, stakeholder satisfaction)[172].

- Requirements do evolve. It is important to separate volatile information from stable information to manage requirements evolution [153]. A requirement is in itself the way of achieving some goal; it is therefore likely that the requirements will evolve towards another possible way of reaching the same goal. Thus, goals are usually more stable than the requirements to achieve them [14].

- Goals give assurance of completeness for the requirements specification by achieving all goals. Thus, providing a criterion for requirements completeness [100] [172].

- Obstacle analysis, in GORE, is helpful in making robust systems by identifying possible ways in which the system may fail and exploring different ways to resolve the problems

early in the validation activity [163].

- Goals provide criteria for requirements pertinence. A requirement is pertinent regarding a set of goals in the domain if its specification is at least used in the proof of one goal [176]. With the help of goal models one can easily see whether a particular goal actually contributes to some high-level stakeholder goal.

3.6 GORE in RE Support

GORE has widely been used to fit different paradigms. It has been used for requirements negotiation, conflict management, risk management etc.

3.6.1 GORE for Conflict Management

Conflicts are not an exception but a rule in RE [57]. Multiple viewpoints and concerns may result in conflicts [125]. Conflicts may be useful, temporarily, for eliciting further information, however they must be resolved [81]. In [55] Easterbrook has proposed a computer-supported framework for capturing multiple perspectives in requirements specification and the resolution of conflicts between them. Lamsweerde et al. [160] have introduced the notion of *divergence* in GORE. In [163] Lamsweerde and Leiter have defined an approach for identifying and resolving the divergence. The nexus of the approach is to manage and resolve the inconsistencies between the requirements at the goal level so that conflicting requirements are not passed on to the system design.

3.6.2 GORE for Negotiation

Conflicts can be resolved through negotiation. According to [58] the main problem in RE is obtaining those requirements which address the issues of all concerned stakeholders. A Win-Win spiral model was used for requirements negotiation to deal with conflicting requirements [27]. [58] has proposed a Win-Win based distributed and collaborative framework for negotiating conflicting requirements and investigating architectural solutions. GORE can be used for identifying the win conditions, conflicts in requirements and obstacles which may be indicative of risks. Goal-oriented requirements engineering has also integrated the process of negotiation to deal with conflicting requirements. [27] proposed an iterative goal-based requirements negotiation model. The iterations for the spiral model of negotiation [27] are

- stakeholders are identified and their goals are elicited *(known as win conditions)*
- conflicts between the goals are captured along with the risks and uncertainties that might be associated with them
- Mutual agreement between goals, constraints and alternatives is achieved through negotiation and reconciliation

3.6.3 GORE for Functional Requirements

GORE can be used to model functional requirements. Functional requirements specify the functions or services of the system. Goals may be referred to as functional concerns or quality attributes [157]. Functional goals result in functional requirements e.g. satisfaction goals is concerned with the satisfaction of users' requirements [163]. A functional goal typically defines a maximal set of desired scenarios. In a GORE process, goals are used to make operational models like use cases, activity diagrams etc.

3.6.4 GORE for Non-Functional Requirements

Non-functional requirements (NFRs) represent software system qualities (e.g. ease of use, maintainability, etc.) or properties of the system as a whole. NFRs are generally more difficult to express in an objective and measurable way. Thus, their analysis is more difficult. GORE has also been used to model the non-functional requirements [73]. The Non-Functional Requirements Framework was proposed in [118] and was later developed in [46]. The NFR framework focuses on modeling and analyzing the non-functional requirements. The framework deals with the following activities: acquiring NFRs for any system, decomposing NFRs, identifying design alternatives for achieving NFRs, selecting the way to operationalize NFRs, and evaluating the decisions. NFRs are represented in the form of softgoals. There are no precise criteria for satisfaction of a softgoal. For softgoals one needs to find solutions that satisfy the goal to a sufficient degree. Softgoals can rarely be fully satisfied. Softgoals can be refined using AND/OR refinements with obvious meaning. Interdependencies between softgoals can be captured with positive ("+") or negative ("-") contributions.

Mylopoulos et al. used a goal-oriented approach for eliciting, specifying and refining the non-functional requirements [118]. The softgoal interdependency graph (SIG) is the main tool for modeling the non-functional requirements. The graph can represent softgoals refinement (AND/OR). As softgoals are being refined, a stage is reached when the goals are sufficiently detailed and cannot be refined further. If a softgoal receives a contribution from a number of other softgoals, then the results of the contribution of every single offspring are combined to get the overall contribution to satisfying the parent goal.

3.6.5 *i**/ Tropos

*i** [173] is an agent-oriented modeling framework which can be used for business process modeling and redesign, requirements engineering, [61], organizational impact analysis, and

software process modeling. Considering the fact that *i** helps to model activities that take place before the system requirements are conceived, it can be used for both early and late requirements engineering phases. The *i** framework models the environment of the system-to-be in the early requirements phase. The framework assists in the analysis of the domain by representing the stakeholders of the system, their goals, and their relationships diagrammatically. The *i** models are used to suggest the new system configurations and the new processes and to assess them to know how well they are able to meet the functional and non-functional requirements of the users in the late requirements engineering phase. *i** is based on the idea of intentional actor and intentional dependency. Actors can be either agents or roles. Agents are systems or humans, with specific abilities. A role is an “abstract actor embodying expectations and responsibilities" [107]. Dependencies between actors are intentional if they appear as a result of agents following their goals.

3.6.6 KAOS

KAOS is a goal-oriented requirements engineering approach having a set of formal analysis techniques. KAOS stands for Knowledge Acquisition in Automated Specification [48] or Keep All Objects Satisfied [164]. KAOS is a framework which has different levels of expressions and reasoning: semi formal to model goals, qualitative for selection of alternatives, and formal whenever required for accurate reasoning [164].

Goal definition in KAOS is “prescriptive statement of intent about some system whose satisfaction in general requires the cooperation of some of the agents forming that system" [156] Goals in KAOS may be indicative of services i.e. functional goals or quality of service (QoS) i.e. non-functional goals.

Goals are refined with AND/OR abstraction hierarchies. Goals refinements ends when goals cannot be further refined or when every subgoal is realizable and can be assigned to an agent. AND refinement of a goal means that the goal is only satisfied when all its offspring goals

are satisfied. Whereas OR refinement relates to a goal that a goal is satisfied if any one of the subgoal is satisfied.

Goals are graphically represented by tree structures as shown in Figure 3.2. In this thesis goals are represented as double edged rectangles. Hexagons are used for agents. An AND-refinement is shown by an arrow with a small circle which connects the subgoals to the main goal. An OR-refinement is represented by different AND-refinement arrows pointing to the same parent goal. A reverse parallelogram represents an obstacle.

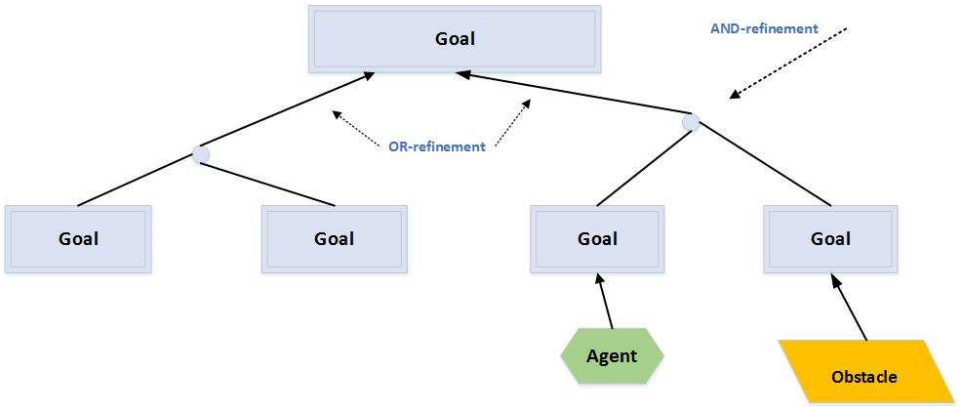

Figure 3.2: KAOS goal model notation

3.6.6.1 Goal specification and classification

Two options can be used to model goals using KAOS, goals can either be specified using natural language or defined formally using a temporal logic formalism. The goal model has a two-layer structure. An outer graphical semantic net layer [31] is used for declaring goals and goal links. An inner layer is used for formally defining goals.

Linear temporal logic (LTL) may be used for formalizing obstacles/goals to enable their behavioral analysis [163]. Some of the logical operators are

$\Diamond$ (sometimes in the future),

$\Box$ (always in the future),

W (always in the future unless),

U (always in the future until).

Goals are classified according to their to the pattern of their temporal behaviour [163]. Goals can be of *Achieve, Avoid, Maintain* and *Cease* types. As an example we formalize the obstacles defined in chapter 5 from the Smart bank case study (see section 5.5.2).

For Obstacle “Substantial amount of sensitive data" (O1) the condition C1 that should never be true is C1= Use the data center in house

The specification pattern for obstacle O1 is:

[if O1 then] never C1

that is, $\Diamond$[O1 $\Longrightarrow$]$\Box\neg$ C1 *(Avoid)*

After prioritizing the obstacle using AHP we found that tactic T1 is best suited for resolving obstacle O1. This is to say that O1 must achieve T1 to resolve O1.

[if [O1 then]always T1]

that is, [O1 $\Longrightarrow$]$\Box$T1 *(Achieve)*

The temporal logic patterns for Cease and Maintain would be:

$\text{Current}_{\text{goal}} \Longrightarrow \text{Target}_{\text{goal}}$ *(Maintain)*

$\text{Current}_{\text{goal}} \Longrightarrow \Diamond\neg\, \text{Target}_{\text{goal}}$ *(Cease)*

In the above patterns $\text{Current}_{\text{goal}}$ and $\text{Target}_{\text{goal}}$ represent current and target conditions respectively

3.6.6.2 Obstacle Analysis

The initial set of goals and requirements tend to be too ideal [163]. It is likely that such over-ideal goals are violated from time to time due to unexpected behaviour of agents. KAOS sup-

ports the detection and resolution of exceptional agent behaviour, which are called obstacles that hinder the achievement of goals [163]. The notion of obstacles was first introduced by Potts [126]. Potts [126] defined obstacles as:

" *Obstacles can be any hindrance in achieving any goal"*

According to Lamsweerde and Leiter [163] the definition of obstacles is:

" *Obstacles capture undesired properties that may prevent the goal from being satisfied"*

In goal-oriented modeling frameworks, obstacles were introduced as a natural abstraction for risk analysis [162] [17]. Obstacles hinder the achievement of goals and unachieved goals may become risks [38]. Obstacle to a goal is a precondition for the non-satisfaction of a goal [163]. Obstacles are unintended risks [20]. Therefore, we liken risks to obstacles. Obstacles can be identified from the goal graph. It is recommended that obstacles be identified from the leaf level goals that are assigned to agents. Potts [126] asked certain questions for identifying obstacles for a particular goal. For example, " Can this goal be obstructed, and if so, when?" According to Anton [13] obstacles denote the reason why a goal failed, scenarios represent circumstances under which a goal may fail.

3.6.6.3 Requirements and Obstacles Completeness

Acquiring complete requirements is of vital importance in RE. The requirements specification is complete if, for a set of goals, all goals can be achieved from the specification in the considered domain [176].

It is important to obtain a complete set of obstacles for every goal (at least for the ones having high priority). As many obstacles as possibles should be identified for the goals especially for the high-priority goals. A set of obstacles O_1,..., O_n can be termed as complete for goal G if the following condition is true

¬O1 , . . .,¬On|= G,

which means that if all the identified obstacles don't occur than the goal is satisfied [163]

3.6.6.4 Suitability of KAOS for Cloud

KAOS aims to support requirements elaboration and modelling. The method identifies and refines high-level goals into subgoals by representing them into a graph structure. According to KAOS, a set of subgoals refines a parent goal if the satisfaction of all subgoals is sufficient for satisfying the parent goal. The refinement of goals continues until the goals can be assigned to agents. This helps in achieving requirements completeness which is otherwise not possible in cloud case where little is known about the features of the cloud service provider. Missing requirements often result from poor risk analysis at requirements engineering time [38]. KAOS helps in identifying the conflicts between different goals [160]. Due to the heterogeneity of cloud users there may be conflicts among different goals. KAOS will therefore be useful in identifying those conflicts. KAOS also helps in identifying and resolving the obstacles [163] which if not resolved may turn into risks. As cloud is designed while keeping in view the generic requirements of the market, a user may also face obstacles in achieving some of his goals. The obstacles may be indicative of risks and must be resolved [179]. Obstacle analysis is a goal-oriented form of risk analysis [38]. KAOS is helpful in not just the identification of obstacles but also in the resolution of those obstacles [179]. Since, our thesis is aimed at mitigating the risks while adopting the cloud services; we argue that KAOS is the right choice for modeling the user requirements and for identifying, analyzing and resolving the obstacles.

3.7 GORE for Risk Analysis

Goal-oriented requirements engineering has been used for risk analysis. In KAOS the notions of *obstacles* [163] and *anti-goal* [159] were introduced in order to analyze the boundary

conditions and the failure condition for a system design. Obstacles can lead to goal failures. Anti-goals are goals which are associated with malicious stakeholders such as a hacker. It can be said that obstacles are unintended risks and anti-goals are threats or intended risks [20].

In [20] a framework for modeling and analyzing the risk during the RE phase has been presented. The framework has adopted the Tropos goal modeling framework and has proposed a qualitative reasoning algorithm to analyze risks throughout the process of evaluation and selection between different options.

[158] argues that requirements completeness is the key issue. Requirements completeness can be achieved by looking at the missing goals/subgoals and obstacle analysis. Incomplete requirements may result in risks. In [38] a quantitative risk assessment technique has been presented which is based on an existing goal-oriented framework. According to Cailliau et al. obstacle analysis is a goal-oriented form of risk analysis. In [38] authors have presented a probabilistic framework for obstacle assessment and goal specification. The probability and the seriousness of the outcome of obstacle occurrence is calculated by up-propagation from the leaf level goals. The computed information can then be used to to prioritize obstacles for obstacle resolution selection.

GORE has also been applied to a case of multi-objective optimization decision problems under uncertainty [97]. Uncertainty makes RE and architecture decisions difficult and may result in significant risks. The authors have proposed to apply multi-objective optimization techniques and decision analysis to provide support for evaluating uncertainty and its impact on risk. A systematic method has been presented that allows software architects to report uncertainty about the impact of options on stakeholders' goals. Monte-carlo simulation have also been used to calculate the consequences of uncertainty [97].

3.7.1 Some GORE Applications

Goal-oriented requirements engineering approaches have been used for different paradigms. GORE has been used for architecture selection, risk analysis and management, self adaptive systems, security and privacy, COTS selection, cloud adoption etc. We discuss some of the GORE applications here.

GORE for Architectures

GORE was also used to model the system architecture to meet changing business goals and for evolving systems [72] [17]. [72] has presented a case study which has highlighted the need for a modeling approach that supports modeling and analyzing how business goals relate to the architectural decision-making process, and how changing business goals result in alternative architectural choices and solutions.

[154] has presented a systematic approach for deriving software architecture from system goals. The approach is based on the KAOS method. It includes qualitative and formal reasoning to achieve the software architecture that meets both non-functional and functional requirements. [94] has defined a three layer reference model which provides the context for discussing the major research challenges which self-management poses. Out of three layers one is a goal management layer which uses the goal-oriented approaches for eliciting and modeling the system goals. [94] has this used goal-oriented approach for defining a three layer architectural model for self-managed systems.

GORE for self-adaptive systems

Goal-oriented approaches have also been used to develop a framework for self-adaptive systems [115]. The proposed methods has used the *Tropos* methodology. Goal achievement conditions were also specified together with their relationships with the environment [115].

[42] has introduced a goal-based modeling approach to develop requirements for a Dynamically Adaptive System (DAS) while taking explicitly into account the uncertainties associated with goal specifications.

GORE for compliance

Organizations need to have a systematic approach to make sure that their business processes comply with certain regulations. A RE framework was presented which could help hospitals to comply with privacy law [67]. The framework uses goals and scenarios were used to help elicit, model and analyze user requirements [67]. The framework was later extended by providing quantitative and qualitative analysis capabilities which allowed organizations to decide what goals have been partially satisfied and which goals have been fully achieved [68].

GORE for security

[156] has proposed a framework based on goal-oriented approach to address malicious obstacles (known as anti-goals) which are set up by the attackers to threaten security goals. The threat tree is derived through anti-goal refinement until leaf nodes reach the vulnerabilities of the system visible to the attacker or anti-requirements which can be implemented by the attacker [159]. [107] proposed a framework that can be used for top-down security requirements analysis, or a bottom-up process that will help evaluate the existing designs. The proposed goal-based analysis technique helps in trade-off analysis of security and other quality requirement [107].

GORE for COTS selection

One interesting application of GORE, which has inspired our work is that of [44] [5] [8], where GORE was used to inform the process of selecting Commercial off the Shelf products matching

user's requirements. A mismatch between user stakeholders' requirements and COTS may result in conflicts. In [7] authors have proposed a goal-oriented framework for dealing with conflicts during COTS selection. Conflicts that arise during the matching process between COTS and requirements can be resolved through negotiation [6]. Although cloud requirements engineering borrows heavily from COTS RE, cloud RE is more challenging due to the ever changing cloud services and user requirements. A COTS product is acquired as a whole by the organization and once it is acquired it remains within the organization. Cloud services on the other hand are instance based and once the job is done the user relinquishes the instance. Besides, in the case of COTS once the organization acquires the product it knows exactly how many features that product has; on the other hand instances are dynamically created according to user needs and every instance is different. Such dynamic nature also makes the risk analysis and mitigation process for cloud very challenging.

GORE for Agent Oriented Programming

[33] demonstrates another use of Goal-oriented approaches in Agent Oriented Programming (AOP) for open architectures that need to change and evolve due to changing requirements. [33] provides a detailed account of Tropos, an agent oriented software development methodology that covers the software development process from the initial set of requirements to the implementation for agent oriented software.

GORE for scalability

[51] has presented a framework for eliciting, modelling and reasoning about scalability requirements using a goal-oriented approach. Obstacles related to scalability are analyzed to assess the likelihood and consequences of the obstacles. It is better to resolve the scalability problems early in the development i.e. at the goal-obstacle analysis level [53]. [52] has described the GORE based approach for eliciting scalability requirements of a large, real-world

fraud detection system.

3.8 GORE for Cloud

Though the fundamental use of GORE resembles the requirements engineering for COTS selection [44] [5] [8] the problem of cloud adoption is far more challenging as we are dealing with “open loop" environments, with dynamic, unbounded and elastic scale where continuous service evolution is the norm. Also, the work done in [44] [5] [8] [7] [6] did not deal with the obstacles/risks encountered. Our work provides a systematic method for identifying, analyzing and mitigating risks in the process of cloud adoption. GORE has been used for modeling and analyzing user requirements for cloud adoption [45]. GORE has also been used to mitigate the risks associated with cloud adoption by resolving the obstacles encountered while achieving some of the user goals [179]. GORE has also been used for prioritizing the risks and risk mitigation tactics using Analytical Hierarchy Process during the process of cloud adoption [178]. It is worth mentioning that due to little knowledge of the cloud services, the requirements engineering for cloud cannot be complete. Negotiation of requirements is therefore heart and soul of requirements engineering for cloud.

3.9 Related Work

GORE has been used for eliciting and refining user requirements for cloud adoption [177]. Eliciting goals is not an easy task [161]. In the cloud case, goals can be elicited from different stakeholders. Cloud providers need to elicit requirements of their customers to develop commercially successful services. Cloud consumers are from different backgrounds and have different requirements. Due to the heterogeneity of the stakeholders and the open and shared nature of the cloud, eliciting goals is a grueling task. A platform named StakeCloud [145]

has been proposed for identifying stakeholders requirements and matching them with the cloud services. The requirements elicited in the form of goals can serve as an input to the platform to help users in identifying a cloud service best suited to their requirements. In case the requirements are not met by the cloud providers, they can be communicated to them as new requirements [146]. The initial set of goals is kept generic in the beginning which are later refined to more concrete goals keeping in view the cloud SLAs.

In the case of cloud it is important that stakeholders engage in negotiation to resolve the conflicting requirements. Due to the generic nature of the cloud services; conflicts can only be resolved through extensive negotiation [177]. Negotiation can be active or passive [177]. In active negotiation a user can negotiate directly with the cloud service provider or through an agent. Whereas, in passive negotiation user can negotiate his/her requirements in light of the readily available information like benchmarks, cloud SLAs, expert judgment etc. [123] has proposed an automated method for negotiating cloud SLAs which resolves conflicting requirements. The method takes inspiration from two economic models - Pareto optimality and Bayesian updating. The Pareto optimal curve results in obtaining the best choices keeping in view the budgetry constraints of the user. The Bayesian updating will require at least one negotiation round to update the new criteria for cloud service selection. The system considers negotiation from the user perspective.

GORE has also been used to identify, analyze and mitigate the risks while adopting the cloud services [178] [179]. The authors [179] have likened risks to obstacles and have proposed some obstacle resolution tactics for mitigating the risks.

3.10 Cloud Migration/ Adoption

Research efforts over the years have looked at the problem of service discovery with runtime mechanisms to inform and optimize the selection e.g. self-managed applications in the cloud

[119] [120] and self-optimizing architecture [119].

There is research on cloud migration [89] which does not involve user requirements for cloud adoption. After performing stakeholder impact analysis through interviews the authors found [89] that there were many benefits of moving to the cloud, however, there were several potential risks as well. Despite having identified the potential risks no work was carried out to mitigate them.

[45] has proposed a goal-oriented simulation approach for cloud-based system design. The approach starts with acquiring goals from different stakeholders and later refining them. The NFR approach has been used for modeling stakeholders' requirements [45]. CloudSim toolkit by Buyya et al. [36] has been used for assessing the impact of design choices on the degree to satisfaction of user requirements. Although the approach demonstrates the way for evaluating and selecting cloud services according to stakeholders' requirements, the approach does not deal with the risks which may be associated with the goals during the cloud adoption process.

[117] has presented a framework for evaluating and selecting a cloud service provider according to the security and privacy requirements. The authors have used goal-oriented approach for identifying the goals, actors, tasks and resources. The authors have admitted that they share our view from [177] that GORE is an important paradigm for engineering requirements for the cloud. The work is however focused on security and privacy requirements and has proposed a framework for screening and selecting a cloud service based on those requirements.

A recent work by Todaran et al. [148] has explored why traditional methods of requirements elicitation are unsuitable for the case of cloud. The authors interviewed 19 different cloud providers to gain insight into how stakeholders' requirements are currently being elicited. The study revealed that most of the cloud providers are using ad-hoc methods for identifying stakeholders requirements. The authors have stressed the need for a cloud specific requirements elicitation framework which takes into account remoteness and heterogeneity of users, and the ever changing cloud services. [147] presents an approach for eliciting user

requirements for cloud service selection by analyzing advanced search queries.

To our knowledge, there has been no research on cloud procurement and adoption from a requirements engineering perspective using the notion of obstacles. The need for such research is timely as there is a complete lack of systematic methodologies, which could help stakeholders screen, match and negotiate their requirements against cloud services' provision and to manage the tradeoffs associated with matches/mismatches of users' requirements and mitigating risks. There may be tradeoffs involved between different requirements. GORE has earlier been used for tradeoff analysis and has proved to be effective to deal with different kinds of tradeoffs [9] [60]. Tradeoffs and negotiation is an integral part of cloud requirements engineering and literature shows that goal-oriented has been helpful in identifying the conflicting requirements and tradeoffs leading to negotiations [5]. Goal-oriented requirements engineering is therefore a promising paradigm where goals can be used to refine, elaborate and negotiate user requirements for mitigating the risks.

3.11 Conclusions

After having a look at the literature we can say that GORE can be used for modelling user requirements for cloud adoption. [177] was the first attempt ever to propose GORE for cloud adoption. In fact prior to that no requirements engineering framework was proposed for cloud adoption. Decisions regarding the selection of cloud service provider were made on ad hoc basis based on the recommendations or on the reputation of the cloud service provider. We also observed that RE should include the risk management process so that users don't get locked in to the wrong cloud. Following our first paper [177], a platform named Stakecloud [145] for matching user requirements with the cloud was proposed. However, this platform did not address the issue of risk management.

Cloud providers also find it difficult to gather user requirements. 19 companies were inter-

viewed to identify the current requirements elicitation methods of cloud adoption [148]. The need for a platform where users and cloud providers can come together was identified. A new approach for eliciting requirements for cloud services by analyzing advanced search queries was proposed [147]. So far, the RE process has been general. Framework proposed are being used for both functional and non-functional requirements.

After our initial work GORE was used for screening and selecting cloud services using the NFR approach [45]. However, even [45] did not try to mitigate the risks associated with cloud adoption. As the mismatches between user requirements and cloud features may be indicative of risks, we felt the need for a risk management framework for the cloud. Later, we proposed a framework using GORE for identifying and mitigating the risks [179]. For mitigating the risks we felt the need for prioritizing the risks in order to know which risks are most important to be mitigated. We used AHP for prioritization. Thus, we prioritized the risks and risk mitigation tactics. Prior to our work there was no RE framework for eliciting, refining and documenting user requirements for cloud adoption. Our method has therefore been helpful in not just acquiring and refining user requirements but also mitigating the risks at the RE phase. The RE activities addressed in all the above work included elicitation, requirements completeness, risk management, obstacle analysis, requirement analysis. Since there was no RE framework for cloud adoption for acquiring, refining and managing user requirements, we believe that our research is timely as there is a complete lack of systematic methodologies which could help users in mitigating the risks in the RE phase of cloud adoption.

CHAPTER 4

Cloud Adoption: A Goal-Oriented Requirements Engineering Approach

4.1 Requirements Engineering for Cloud Adoption

Let us assume that University of Birmingham (UoB) wishes to outsource its email services to the cloud. This is a crucial decision as the university has plenty of specific requirements, which cloud providers need to satisfy. For example, the UoB must continue to conform to the data protection act of the United Kingdom. It must also conform to the university regulations for processing institutional data. It must avoid any risks or liability due to breaches in data protection and confidentiality. Consequently, the UoB remains apprehensive about the implications of outsourcing. The cloud provider must also satisfy UoB's functional requirements. The cloud provider must also satisfy UoB's non-functional requirements such as reliability, safety, 24/7 availability, response time in peak usages and unbounded scalability of the e-mail service provision. Furthermore, UoB should also analyse the cost versus benefit of the decision in moving to the cloud compared to in-house provision. The selection should also address UoB's requirements for maintainability, future upgrades along with the need for reducing future operational cost. The decision involves concerns of many users regarding cultural shift with cloud adoption. UoB would strive for systematic evaluation for all the tradeoffs and risks involved in considering specific clouds. As the decision concerns many users and would result in a cultural shift in practice; UoB would strive for systematic evaluation for all the tradeoffs and risks while considering specific clouds.

4.1.1 Requirement Elicitation

The requirements for cloud adoption should not cover all the details in the initial stage. The initial stages of requirements engineering for cloud should be flexible enough to permit negotiations and further refinements owing to the evolving needs of the user. One of the reasons to keep requirements generic is that we don't want to eliminate promising cloud service providers at the initiation. Cloud service providers cannot meet all the requirements of the user; it is therefore, unwise to spend time and effort on eliciting a comprehensive set of requirements that might not eventually be satisfied. To start with, we should acquire

the core requirements which should not be rigid; they should allow future negotiations and elaboration. As soon as the basic requirements are elicited, the search for a potential cloud service provider can start; next requirements can be elicited with the evaluation of the potential cloud providers and services provisions. UoB can begin a search for prospective cloud service providers after eliciting the core requirements (e.g. security of the data, backups, availability, core functional requirements for the e-mail service etc.)

4.1.2 Requirement Analysis and Negotiation

The requirements analysis is an interactive and iterative process, where the refinement of the organisation's requirements is driven by the availability of the cloud service providers and the accessibility of information related to the provided features of service usages, the quality of service provision and other terms and conditions etc. Information documenting features and service provision of different clouds are accessible through various sources. Examples include and not limited to the cloud white papers, SLAs, available benchmarks e.g. CloudHarmony, the Internet, reviews, evaluation and recommendations of users, experiences, cloud marketing representatives and the like. By looking at the features provided by the cloud service providers, the user/organisation may come up with a new set of requirements that were not initially identified. The examination of the cloud service providers' features is a better technique to refine and understand how high-level requirements of the stakeholders can be actually satisfied. As cloud services and SLAs are designed to satisfy the generic and wider requirements of the market, some users' requirements may not be satisfied. Users should, therefore, be prepared to engage in an extensive process of requirement prioritization and negotiation. Negotiation can be either active or passive which have been earlier defined in Chapter 2. It is a common practice for some cloud service providers such as Amazon to allow negotiation of SLA terms and conditions [10]. In order to successfully negotiate their requirements, users have to perform continuous tradeoffs and risk analysis in satisfying a particular requirement against the limitations of the available cloud providers. As an example, UoB has to abide by the Data Protection Act of the United Kingdom (UK) but the cloud service

provider's storage location is outside United Kingdom. UoB looks at the other possible storage locations and agrees to store data in any European Union member country. This will allow UoB to use cloud services while not violating the Data Protection Act of the UK. In other words, cloud's available features may help in refining and informing more realistic set of user's requirements. Moreover, UoB can use this phase to investigate and possibly negotiate about the location of data, access controls, backups and archiving mechanisms, QoS provision and so forth.

4.1.3 Requirements Evaluation

In the process selecting candidate cloud service provider(s), the evaluation of requirements is a continuous process of cloud evaluation and requirements negotiation, where the final set of requirements will undoubtedly be a compromise and fair reflection between what users want to achieve and what is offered by the cloud service provider. The purpose of requirement evaluation is to short list the cloud candidates and to ignore some service providers. The requirements evaluation phase will try to screen candidate cloud service providers for further evaluation.

4.1.4 Requirements Documentation and Management

The requirements documentation can be regarded as a semi-formal prebinding connection between the user and the cloud service provider; it acts as an informal contract for cloud adoption and users' requirements/cloud feature negotiation. In the process of cloud service provider selection, the requirements documentation will serve as the basis for cloud evaluation. Once UoB has refined its requirements, it will prepare a document wherein all the requirements will be defined. This set of requirements will be agreed upon by the cloud service provider and UoB. Requirements are documented to ensure that the cloud service provider gives what is required by the user. The requirement documentation will help in producing an SLA addressing the specific requirements of a user. The documentation will

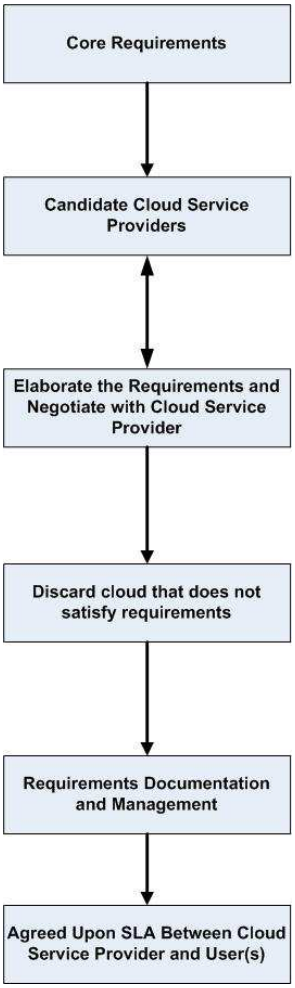

Figure 4.1: Requirements Engineering steps for Cloud Adoption

also serve as a basis for deciding if any SLA has been violated by the cloud service provider. In case of a cloud the requirements engineering should be a continual process involving tradeoff analysis, negotiation and risk management. The reason behind doing so is that if the cloud service provider brings any changes to its system then the user will have to update the system according to the changes. For example this will require all the users of UoB's email service to upgrade their operating system to gain access to their emails. Figure 4.1 shows requirements engineering steps for cloud adoption.

4.2 Cloud-Based Goal Oriented Requirements Engineering

The specification of users' requirements is the first step towards selecting an appropriate cloud service provider. Boehm estimates that late corrections of requirements errors could cost 200 times as much as corrections during requirements engineering phases [26]. Though Boehm's argument is related to the case of projects and software developed for relatively defined set of requirements, it could be argued that late rectification of unwise selection may lead to higher cost, which could be disproportional to the benefits of using the cloud and will place such investment at risk. Our approach starts with high-level goals. Initially when any organisation decides to move to the cloud, the goals are general expression of the requirements, whether these are strategic requirements, functional and/or non-functional. The higher level goals (such as secure payment transactions) are more stable than low level ones (such as the use of Secure Socket Layer (SSL) Technology). Defining goals is the first step of requirements elicitation in software engineering. Goals have been earlier defined in section 3.4.1 To make a sensible decision for selecting a cloud service provider, businesses require a methodology which could help them make a choice. In conventional software development, the requirements engineering basically consists of eliciting stakeholders' needs, refining the goals into non-conflicting requirements, followed by validating these requirements with stakeholders [124]. The main objective of the requirements engineer is to ensure that the requirements specifications meet stakeholders' needs and represent a clear description of the cloud services that are to be adopted. Stakeholders' requirements play a deciding / leading role in cloud service provider selection. Before a final selection of the cloud service provider is made, goals would have to go through certain steps as shown in Figure 4.1. The lifecycle defined in Figure 4.3 is based on the steps involved in Requirements Engineering for Cloud Adoption as shown in Figure 4.1.

4.2.1 Acquire and Specify Goals

Requirements act as criteria to compare and evaluate cloud service providers. The goals can be divided into three categories (i) strategic or business goals, (ii) high level or core goals,

(iii) and low level or operational goals. Strategic goals are concerned with the survival of the enterprise/business. Assume that UoB wants to drop the cost of email operations by 20% over the next five years; this is the strategic or business goal of UoB. High Level goals are nothing but core goals described in requirements engineering phase for cloud adoption. Suppose UoB desires that the cloud should be able to handle 500 users during the peak hours and provide a high level of security - these are high level goals to be achieved. First high level goals will be acquired from UoB that are refined into more objective sub-goals until they reach the level of operational goals. Operational goals can interact among each other. Operational goal can be encryption of data, password protection, and backup mechanisms etc. Sometimes, operational goals need to be assessed and evaluated against high level core goals.

We will have to define the acceptance level for each goal as these cannot necessarily be absolutely met; instead they are satisfied to a certain degree within acceptable limits [118]. The acceptance interval ranges from target level i.e. the optimum value that users consider to be fully satisfied, to the worst level, i.e. the lowest value of the acceptable interval in which the goal starts to be considered unsatisfied.

Guidelines to acquire and specify goals

1. **Acquire Goals**

   To start with, we acquire generic requirements from the users which are flexible and can be negotiated against the features of a cloud service provider.

   Start with acquiring very generic goals. Goals can be acquired using different methods. Goals can be elicited through scenarios, interviews of the stakeholders, documentation of the current existing system etc. It is also essential to understand the domain.

2. **Assess cloud**

   Assess features of the cloud service provider from sources like cloud SLAs, benchmarks, whitepapers, market reviews etc.

   After assessing the features of the cloud service provider through different sources

users may end up with having new requirements. Some of the requirements may be discarded after examining the features while some of the new requirements may be added.

3. **Specify Goals**

   Elaborate and refine the goals in the light of cloud features.

   The refinement of goals should be performed with respect to features of the cloud services. Define the relationship between the goals using AND/OR decomposition. During the elaboration process there will be a point where goals can no longer be refined and will have to be assigned to different agents for operationalization. It is important to understand that agents may not be able to satisfy the goals fully and therefore users should define the acceptance level for the goals.

For example UoB wants the response time to be 10 seconds ideally as against 15 seconds which is unacceptable. UoB has set the acceptance limit to 12 seconds which is within the worst and target range. The goals need to be prioritized once they are specified, users have to engage in an extensive prioritization process in order to distinguish core goals (i.e. critical needs that should always be satisfied) from desirable goals (i.e. the ones that could be traded off).

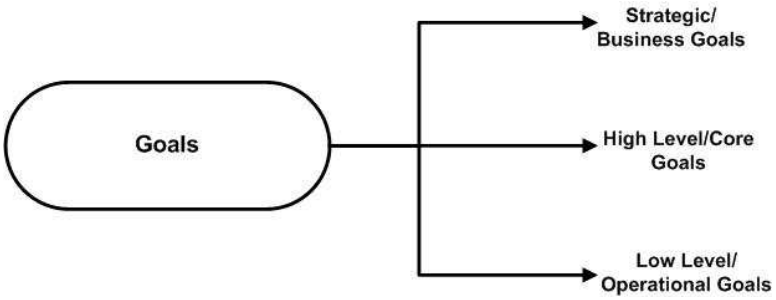

Figure 4.2: Categories of goals

4.2.2 Assess Features of the Cloud Service Provider

Requirements are elicited in the form of goals. We start with acquiring the high level goals. At the initial stage goals are kept generic so that the search for a cloud service provider is not limited by unnecessary constraints. Once the generic goals have been defined the search for potential cloud service providers will start. At the initial phase of cloud service provider identification, it should be ensured that the cloud satisfies the critical goals; total cost of ownership match the available budget, reputation of service provider with other users is compared and that the service provider is willing to participate in a collaborative partnership etc. Once we identify the clouds which are able to satisfy the core goals then we can start evaluating them. We may evaluate one cloud or more than one cloud which satisfy the high level goals and are ready to negotiate. We perform the matching between user requirements and the features of the cloud service provider. If there are any mismatches between the requirements and the cloud features then they may be indicative of risks. The mismatches should be analysed for any possible risks. Risks should be mitigated at this stage so that they don't pass on to the final system. Risk mitigation may result into new requirements. Once the risks have been mitigated and the cloud is able to sufficiently satisfy user requirements, the user may decide to adopt the cloud.

The UoB may arrange demonstration sessions to ascertain how well cloud satisfies the specified goals. The assessment of cloud service provider's features involves a great deal of uncertainty, where key information to assess the satisfaction of goals can be difficult to obtain. To ensure possible accuracy of the assessment process, each observed functionality must have a confidence degree associated with it. This confidence degree is based on well justified arguments and evidence that a desired functionality (i.e. operational goal) is sufficiently satisfied. The highest confidence degree is obtained when the goal satisfaction is verified while the lowest one happens when the goal satisfaction is informed. After selection of the cloud service provider the UoB may ask for a trial period which could help verify that a cloud satisfies the requirements and UoB does not have to sign a contract at the beginning without experiencing the services.

4.2.3 Perform Matching

This phase involves gathering sufficient information about the cloud service provider and its services in order to assign the satisfaction scores to operational goals. Based on the results of matching, the evaluation team can aggregate individual satisfaction score (i.e. how the cloud satisfies each operational goal) into global satisfaction scores (i.e. how the cloud satisfies the set of operational goals). Therefore, it is possible to compare cloud service providers and then, inform the decision making process. The matching process involves analysis of cloud service provider satisfaction and further discussion with users to determine whether or not a particular cloud sufficiently satisfies their needs. The matching of a particular cloud is considered satisfactory if the cloud satisfies the operational goals within an acceptable range. The degree of satisfaction can come from individual forums like CloudHarmony [1]. Some clouds may offer the user a trial period where the user can see the degree of satisfaction of the cloud. For example, for UoB, goal is satisfied if the response time is less than 12 seconds.

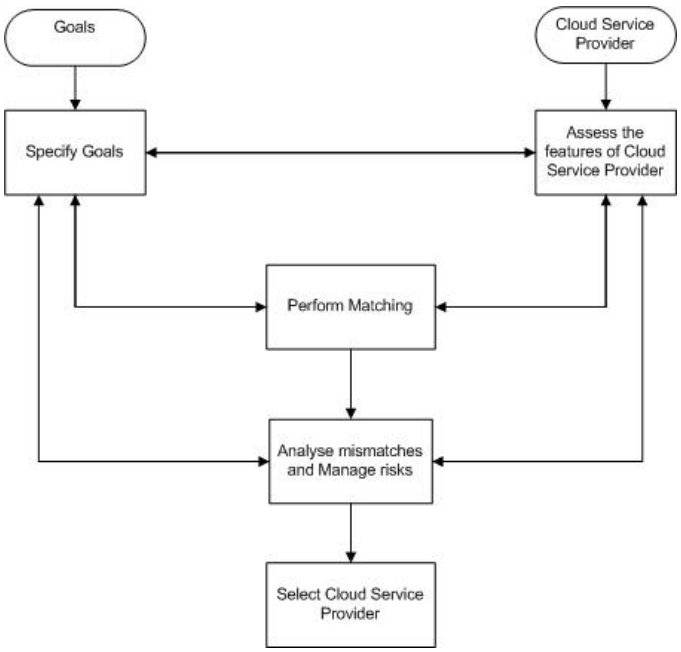

Figure 4.3: Steps for Cloud Adoption

4.2.4 Analyse Mismatches and Management of Risks

A systematic approach is needed for the evaluation team to understand the effects of mismatches, analyse conflicts between goals, explore tradeoffs and manage risks. In the context of cloud service provider selection, risks are defined as unacceptable outcomes, generally caused as a result of conflicting goals and mismatches. Given that mismatches represent non-adherence of cloud to operational goals and conflict arises when satisfying one goal damages the satisfaction of another goal, we deduce that risks arises when the loss caused by unsatisfied goal is intolerable. A fundamental issue in handling mismatches is the capacity to systematically structure tradeoffs. The tradeoff analysis forms the basis of risk management strategy. The objective of a risk management strategy is to understand and handle risk events prior to their conversion to threats. Risk management helps in successfully selecting and integrating the chosen cloud service provider. The risk management process has three steps: risk identification, risk analysis and risk mitigation. Risk mitigation action may cover options such as: change goals, negotiate cloud service features, or choose other alternatives. There is no one solution for all the risks; therefore, the decision for the best risk mitigation strategy relies on the judgments and experience of the evaluation team. This process is similar to the process *Elaborate the Requirements and Negotiate with the Cloud Service Provider* defined in Figure 4.1. In this phase UoB will match its goals with the cloud service providers' features and decide which cloud service provider best fulfills the user requirements.

4.2.5 Select Cloud Service Provider

The ultimate objective of this phase is to choose the “optimal cloud". By optimal selection we mean that the selected cloud not necessarily needs to be optimal but satisfying. A satisfying cloud is one that sufficiently satisfies the set of goals defined by stakeholders, where as an optimal cloud aims to maximize the satisfaction of goals. User will assess features of all the cloud service providers who are able to satisfy the core goals. For example, if three different cloud services are able to satisfy the core goals then the user will assess all three cloud services

and then decide which one can be adopted. Factors to assess the merit of each cloud service provider for selection are value, cost, and risk. Cost represent the monetary investment, time and effort. The overall merit of a cloud based system is the combined measure of the value, cost, and risk. Value refers to the level of satisfaction of the stakeholders if a given alternative successfully satisfies the desired goals. Notably this is related to the priority stakeholders have assigned to goals. The risk is assessed using goal mismatches, conflicting goals and involves risk scenario elements. UoB during this phase will identify the cloud service provider that satisfy the user goals within the budget constraints.

Guidelines for initial rejection of the candidate cloud services

1. Is the cloud able to meet the critical goals which cannot be negotiated?
2. Is the cloud under the budgetary constraints of the organization?
3. Have there been frequent SLA violations by the cloud providers previously?
4. Has the cloud been used for similar purposes and how was the response of the customers?
5. Is there enough information available about the cloud services and are there any benchmarks available to keep a tab on cloud services for any SLA violations?
6. Is the cloud service able to work in the existing system or does the organization have to make significant changes to their existing system to adopt cloud services?

4.3 Discussion

Cloud computing has gained popularity among businesses in recent years. Users are as excited as nervous about using the cloud. They are excited as most of the services provided by the cloud are low cost and readily available. At the same time, in spite of many promises by the

cloud service providers, users remain concerned about the general risk associated with the adoption of the cloud such as security and privacy of the data held, processed and exploited. We have proposed a framework which would help in addressing the genuine concerns of the cloud users. We looked at the case study of a cloud service provider and found that there were indeed many risks associated with cloud adoption. We advocated the necessity of a systematic methodology, for cloud adoption. The methodology can help the users in refining their requirements and negotiating with the cloud service provider while using a goal oriented approach. Negotiating and changing the SLA is situation dependent. For large investments a cloud service provider may engage in negotiations. There is an interesting clause in Amazon Web Services Customer Agreement which prohibits the user from disclosing the agreement for three years after the end of the agreement: “the nature, content and existence of any discussions or negotiations between you and us or our affiliates" [10]. This statement is clearly indicative of some background negotiations between the cloud service provider and user. Our approach will help the users to identify the conflicts between the requirements and to reduce them. There might be certain obstacles involved in achieving some goals. Obstacles can be any hindrance in achieving any goal or undesired properties [126]. Obstacle analysis needs to be done in the initial phase of requirements engineering i.e. at the goal level [163]. Obstacles can also be regarded as an expression of risks. Our aim is to manage the obstacles in achieving our goals. We intend to evaluate our framework by looking at a case study.

4.4 Conclusions

We have defined the steps involved in the requirements engineering phase for cloud adoption. This chapter contributes to a novel lifecycle, which aims at providing systematic guidance for an organisation evaluating the choice and risks in moving and adopting a cloud. We have used a goal oriented approach for eliciting and modeling the requirements of the user. As the cloud services evolve so do the user requirements. Since goals are more stable as compared to requirements [14] therefore it is logical to model the requirements using the GORE paradigm.

We have presented systematic guidance for the identification and evaluation of cloud service providers. The evaluation process concludes in the selection of the best cloud service provider available. The key phase of the method is the matching phase where mismatches between the requirements and cloud service providers' features are identified. The analysis of the mismatches may inform the existence of risks. Our approach advocates risk mitigation in the early stages of cloud adoption. We have used GORE for mitigating the risks. In KAOS the notion of obstacles have been used in order to analyze the conditions under which a goal may not be achieved. The mismatches between cloud features and user requirements may be due to certain obstacles in the achievement of goals. We have used the goal/obstacle analysis to analyze and mitigate the risks in the cloud adoption process which will be discussed in chapter **??**. The approach also tries to manage the tradeoffs involved with the cloud adoption. The Service Level Agreements presently are too static and non-negotiable. The SLAs do not address the individual needs of every user. This framework would help the cloud service provider and the user to negotiate their requirements and cloud features beyond static SLAs. It would help making SLAs that are specifically designed for a particular organisation.

CHAPTER 5

Using Obstacles for Systematically Modeling, Analysing and Mitigating Risks in Cloud Adoption

If you don't invest in risk management, it doesn't matter what business you're in, it's a risky business

Gary Cohn

5.1 Introduction and Motivation

We have discussed in chapter 2 that cloud RE is more challenging as compared to traditional RE due to the ever changing user requirements and cloud features. Cloud services are usually not tailored according to the individual needs of the customers. We believe that matching user requirements with the features of a cloud service provider will unveil potential risks associated with the adoption of a particular cloud service. These risks if not mitigated at the adoption phase may expose users to significant losses. We therefore believe, that risk management should be part of the requirements engineering for cloud adoption.

We looked at the case study of a leading cloud service provider- referred to as *Indus* which has been earlier defined in section **??**. After looking at the case study we realized that cloud service providers try to transfer the risks to their customers through non-negotiable SLAs. Hence, it is important for the cloud users to screen the cloud according to their requirements to reveal the associated risks before adoption to avoid getting locked-in with the wrong cloud. We proposed a lifecycle based on goal-oriented requirements engineering for cloud adoption in chapter 4. This chapter is an extension of our previous work presented in chapter 4. We have refined the cloud adoption lifecycle 4.3 and have introduced the notion of obstacles. Sometimes there are hindrances to the achievement of goals/requirements called obstacles. The unfulfilled requirements due to obstacles may eventually lead to risks. It is important to resolve these obstacles in order to mitigate the risks. We have also defined some concrete and abstract obstacle resolution tactics in this chapter.

This chapter is structured as follows. Risks that were identified from different cloud service provider's SLAs are presented in section 5.2. In section 5.3 we introduce goal-oriented requirements engineering for the process of cloud adoption. We define obstacles in section 5.4 and argue that obstacle analysis should be part of the lifecycle for cloud adoption process. We have modelled a case study from two different perspectives (user and cloud service provider) using a goal-oriented approach in section 5.5. From the user perspective we have modelled the case study of "Smart Bank" which has decided to adopt a cloud [62]. We have identified and resolved the obstacles that may hinder the adoption of cloud by the Smart Bank. From the

cloud service provider's perspective we have modelled the case study of "Indus" which is one of the leading cloud service providers. After careful look at the Indus Web Services Customer Agreement and Indus whitepapers we found several risks associated with the adoption of cloud. We have proposed some obstacle resolution tactics which could help in mitigating the risks associated with cloud adoption in section 5.6. Section 5.7 defines a method for prioritizing goals using the utility theory. We discuss the open problems and conclude the chapter in section 5.8.

5.2 Risks Identified from Different Clouds and Their Implications

Risks that were identified by looking at the SLAs of different cloud service providers are shown in Table 5.1. The information presented in Table 5.1 has been collected after inspection of SLAs and white papers of several leading cloud providers such as Amazon Web Services, Google AppEngine and Microsoft Azure. Risks could be classified into various areas of concern (e.g. network, environment etc.) For each identified risk, we provide an impact assessment of the occurrence of the risk. We believe that the identified areas, potential risks and impact assessment is representative of the state-of-the-art in cloud service provision. Although, cloud service providers have tried to win the confidence of the users by either publishing their white papers, web service agreements or by providing the details regarding their cloud service on their website, most of the cloud service providers have failed to take the responsibility for the users' content if the provider is unable to meet its promises to the user. The SLAs of most of the cloud service providers absolve them from any responsibility should anything happen to users' content. Signing a contract with such cloud service providers can be a risky business. It is evident from Table 5.1 that many risks could eventually lead to financial losses for a business. To minimize the risk of being locked in with a cloud provider who is unable to meet the needs of the users; the proposed framework informs cloud provider selection on a systematic and sound framework.

Table 5.1: Risks identified from cloud service providers SLAs

Area	Description of Risk	Impact Assessment
Network	IP Spoofing	Data leakage
	Port Scanning	Financial losses
	Packet Sniffing	Customer trust
Environmental	Fire	Hardware damage
	Electrical failure	Customer trust
	Temperature inside the data center	Business reputation
		Huge financial losses
Technical	Integration risk	More investment on company's infrastructure
	Insecure interfaces and APIs	Huge financial losses
	Shared technology issue	Data leakage
	Weak encryption of data in transit	Company's reputation
	Incomplete data deletion	Customer's trust
	Resource exhaustion	Service delivery
Physical Access	Unauthorized access to the data center	Financial losses
	Malicious insider	Customer's trust
		Business Reputation
		Data leakage
		Hardware damage
Business Risks	Compliance challenges	Industry certifications
	Natural disasters	Company reputation
	Data loss or leakage	Customer trust
	Unknown risk profile	Closure of business
	SLA clauses containing business risks	High financial losses
	Incomplete data deletion	Going bankrupt
	Long term viability	Company's secret made public
	Interoperability	Service delivery
	Resource exhaustion	
Legal	Risks from change of jurisdiction	Financial losses
	SLA clauses containing business risks	Customer trust
	Unknown risk profile	Business reputation
	Data recovery and investigation	Service delivery

5.3 Cloud-Based Goal-Oriented Requirements Engineering

Our framework uses goal-oriented requirements engineering for specifying user requirements. Our approach starts with acquiring high level goals and later elaborating them to low-level goals. Section 4.2 describes the cloud-based GORE. In this chapter we introduce the notion of obstacles for mitigating the risks in the cloud adoption process. Figure 5.1 shows the cloud adoption lifecycle which takes into account the obstacles to goals.

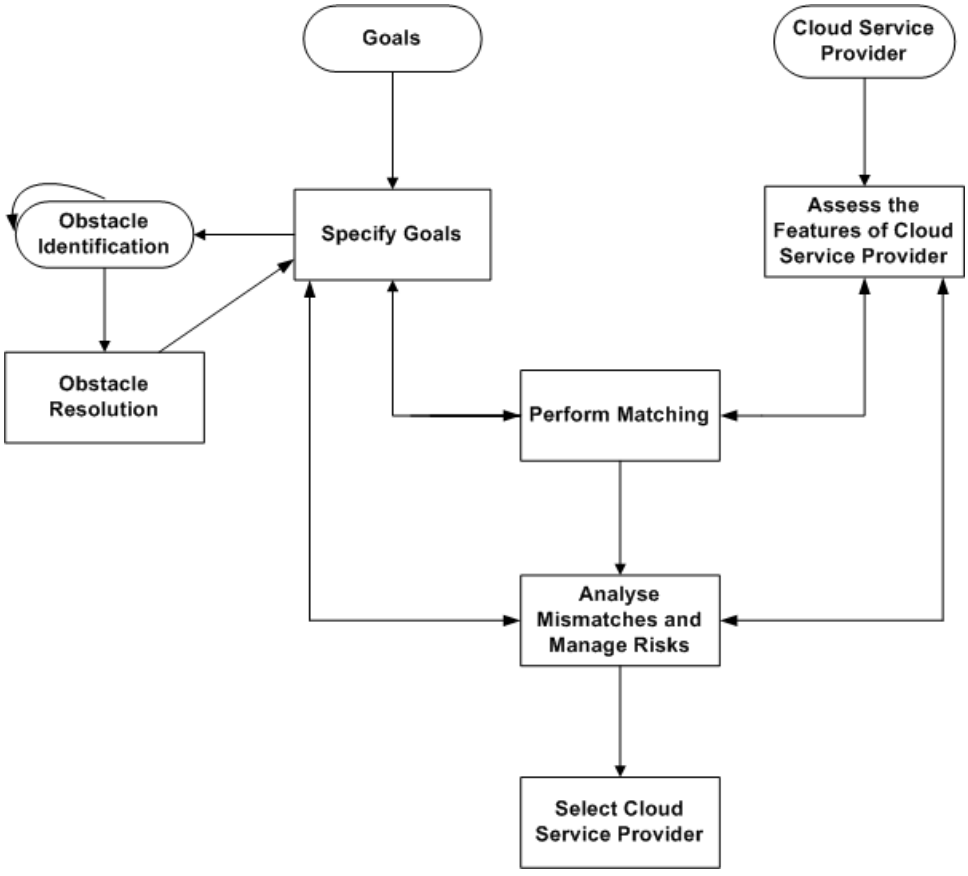

Figure 5.1: Lifecycle for Cloud Adoption

5.4 Obstacles for Mitigating Risks in Cloud Adoption Process

We first proposed a methodology which could help users to adopt cloud services [177]. The methodology is grounded in Goal-Oriented Requirements Engineering. The initial work did

not provide the means to resolve obstacles for achieving goals. Requirements engineering is concerned with the elicitation of high-level goals that are to be achieved by the system-to-be. The high-level goals are refined into low level goals which can be operationalised. Classically in GORE we operationalise goals by assigning them to agents such as humans, devices and software (in our case, the agent can be either a cloud service provider or any particular service). Sometimes the requirements engineering process results in goals and assumptions about the agent behavior which are too ideal; some are unlikely to be satisfied by the system. We present techniques to resolve the obstacles in achieving goals. We are defining a number of abstract and concrete techniques for resolving obstacles which could mitigate risks in the cloud adoption process.

Steps for cloud adoption according to figure 5.1

1. **Acquire goals**

   To begin with, we start by acquiring high level goals from the user which are flexible and negotiable. The goals are kept generic in the beginning so that the search for a cloud service provider is not limited by unnecessary constraints.

2. **Identify the cloud(s) for further evaluation**

   If a cloud cannot satisfy the initial set of generic goals then it's best to reject that cloud in the beginning. The cloud which can be considered for further evaluation should be under the budgetary constraint of the user and should be able to meet at least the most critical goals that cannot be negotiated.

3. **Specify goals**

   Once high level goals have been acquired, they can be further refined into realizable goals. They can be refined in light of the cloud features/SLAs. The relationship between the goals can be defined using the AND/OR decomposition.

4. **Assess features of the cloud service provider**

Assess features of the cloud service provider from different sources like benchmarks, SLAs, market reviews etc. Some of the requirements may be discarded after examining the cloud features while some of the requirements may be added.

5. **Perform matching**

   Perform matching between the features of the cloud service provider and user requirements. If there are any mismatches they may be indicative of possible obstacles and should be resolved.

6. **Analyse mismatches and manage risks**

   After matching features of the cloud service provider with user requirements, we may find some requirements that the cloud service provider cannot fulfill. It is important to analyse those mismatches and see if they result into any kind of risk. If there are any risks due to the mismatches between user requirements and cloud features then it's best to mitigate them at the requirements engineering level than passing it on to the final system.

7. **Obstacle identification and resolution**

   The obstacles can be generated during the specify goals phase or during the matching process. The obstacles should be recursively refined. The obstacle resolution tactics should be used for resolving the identified obstacles.

8. **Select cloud service provider**

   Select a cloud which is able to satisfy user requirements and can resolve the obstacles encountered during the evaluation process of the cloud service provider.

5.4.1 Obstacle Analysis

The first sketch of goals tend to be too ideal [164]. Such over ideal goals can be violated from time to time due to unexpected behavior of agents. In KAOS, such exceptional behaviors are called obstacles to goal satisfaction. Obstacles were first proposed by Potts in [126] . He

identified obstacles for a particular goal by asking different questions. For example: “Can this goal be obstructed, if so, how?" According to Anton [13] while obstacles denote the reason why a goal failed, scenarios determine concrete circumstances under which the goal may fail. KAOS provides well-developed methods for identifying and resolving obstacles. Lamsweerde et al. [163] recommend that obstacles be identified from leaf-level goals. Once identified, obstacles can be refined like goals (by AND/OR decomposition). Obstacles can lead to risks. Obstacle resolution will eventually result in minimizing potential risks.

5.4.2 Obstacles in the Requirements Engineering Process for Cloud Adoption

We found that there are some obstacles in achieving goals. We have therefore refined the lifecycle for cloud adoption (Figure 4.1) by introducing obstacle identification and resolution in the lifecycle (see Figure 5.1). During the “Specify Goals" phase the goals are refined and obstacles are generated. Obstacles can be repeatedly refined. The generated obstacles are resolved which results in updating goals. New goal specifications may result in a new iteration of goal refinement and obstacle identification. It can therefore be said that obstacles analysis and resolution has to be a continual process. Figure 5.1 shows the life cycle for cloud adoption. An obstacle from the perspective of a cloud user can be defined as any hindrance in achieving any goal or the desired properties as set out by the cloud user. Obstacles obstruct the goal such that when the obstacle is true then the goal may not be achieved . Suppose a social networking website hosted on cloud wants 24/7 availability. If there is a power outage at the data centre of the cloud service provider then the service may not be available and hence power outage will be an obstacle in achieving the goal of 24/7 availability. We briefly describe the steps involved in the lifecycle shown in Figure 5.1.

5.4.2.1 Acquire and Specify Goals

Requirements act as criteria to evaluate cloud service providers.We start by acquiring the core goals and then refine them later. Section 4.2.1 sufficiently describes this step.

5.4.2.2 Obstacle Identification

During the *specify goals* phase obstacles are generated. The obstacles can be recursively refined. Obstacles are identified from the goal graph. It is suggested that obstacles be identified from the terminal goals that are assigned to agents. It is essential that the set of identified obstacles is complete for every goal (at least for high-priority goals). To resolve obstacles we would like to identify as many obstacles as possible for the goals; particularly for high-priority goals. A set of obstacles O_1,..., O_n is complete for goal G if the following condition is true.

$\{ \neg O_1, ..., \neg O_n \} \models G$

This condition means that if none of the identified obstacles occur than the goal is satisfied. Figure 5.2 and Figure 5.3 show the obstacles identified from the case study.

5.4.2.3 Obstacle Resolution

Obstacle resolution tactics are applied to resolve the identified obstacles. The selection of a specific obstacle resolution tactic depends on the likelihood of the obstacle occurrence and the impact of such an occurrence. The goal/obstacle loop may terminate once the obstacles that remain are thought to be acceptable without applying any resolution tactics. Table 5.2 shows some obstacle resolution tactics in the process of cloud adoption. We have applied these tactics to resolve the obstacles in the Smart Bank case study in section 5.6.

5.4.2.4 Cloud Service Provider

After acquiring the generic goals, the search for a potential cloud service provider will begin. Initially goals are kept generic so that the search is not limited by unnecessary constraints. If the cloud is able to satisfy the critical goals and the amount required to adopt cloud is within the budget then the goal tree can be further elaborated and cloud features may be screened.

Table 5.2: Obstacle resolution tactics for cloud adoption

Category	**Short Description**	**Obstacle Resolution Tactic**	**Example**
Obstacle Prevention	Avoid The Obstacle	Tactic 1: Add another goal to prevent obstacle Tactic 2: Improve/Negotiate SLA.	Tactic 1: Encrypt data (add goal) before sending it to the cloud to achieve security of the data. See figure 5.6 where obstacle encountered in Smart Bank case study has been resolved by adding a goal i.e. to encrypt sensitive data using Trusted Platform Module when storing it on the cloud. Tactic 2: Negotiate SLA for storing the data in the desired location. See figure 5.7
Cloud Service Substitution	Assign the responsibility of obstructed goal to another cloud	Tactic 3: Transfer the operationalization of the goal to another cloud. Tactic 4: Split goal handling among multiple clouds	Smart Bank wants its system to be available 24/7. However, a service may be unavailable during peak hours. A different cloud may be used during peak hours to handle the extra load. See Figure 5.8 and figure 5.9 Federated cloud can be used for different applications to match business needs. See figure 5.10
Goal Weakening	Weaken the obstructed goal specification	Tactic 5: Weaken goal objective function Relax requirements Relax the degree of satisfaction	Response time required for an email service can be increased to a few seconds if the cloud provider is unable to guarantee the desired response time. See figure 5.11

5.4.2.5 Perform Matching

Perform matching between the features of cloud service provider and user requirements. The information regarding features and service provisions of different clouds can be accessed through various sources like cloud white papers, SLAs, available benchmarks, the Internet, reviews, evaluations and recommendations of users, experiences etc. Section 2.3.2 describes the matching process in detail.

5.4.2.6 Analyse Mismatches and Manage Risks

Mismatches between cloud features and user requirements may be indicative of risks. It is important to devise a risk management plan at this stage. Section 4.2.4 describes the process of analysing and managing risks.

5.4.2.7 Cloud Service Provider Selection

The objective of this phase is to select a cloud which is best suited to the user keeping in view the budgetary constraints. It is important to select a cloud which not only satisfies user requirements but also minimizes risk occurrence. The risk is assessed using goal mismatches, conflicting goals and involves risk scenario elements. Section 4.2.5 sheds some light on this phase.

Guidelines for identifying the risks

1. Are there conflicts and tradeoffs between different requirements? Can any of them be relaxed for the other?
   E.g. Performance may be affected when encrypting large amount of data online.
2. Are there any mismatches between the user goals and features of the cloud service? Mismatches may result in risks. Some of the mismatches may be negotiated and user and cloud may agree upon a solution
   E.g. User may agree upon 95% availability of the cloud instead of 99%
3. Identify the obstacles to the fulfillment of the requirements. Obstacles may be indicative of possible risks.
   E.g. incomplete data deletion by the cloud provider may be a risk to security.

5.5 Case Study

We have modelled two perspectives (the cloud service provider and cloud user) of a case study. Figure 5.2 and Figure 5.3 show the goal tree of the obstacles that have been identified in the case study. For the purpose of this case study we have adopted the following conventions. Double edged rectangles show the goals, parallelograms are the obstacles in achieving some of the goals and hexagons represent an agent. An AND-refinement is represented by an arrow with a small circle connecting the subgoals contributing to the parent goal. An OR-refinement is graphically represented by multiple arrows pointing to the same goal. The objective of modelling two different perspectives of a case study is to show obstacles at the service provider and the user level. Cloud service provider may have claimed high security in their cloud but a close look at their whitepapers revealed that the risks associated with the adoption of such cloud are imminent. After looking at the Indus Web Services Customer Agreement and Indus white papers we identified many risks associated with adoption of Indus cloud. For the purpose of simplicity we have not modelled all the risks identified. The goal graph for Indus informs users of the possible risks that may occur due to adoption of this cloud. The goal tree of Smart Bank helps us in introducing some obstacle resolution tactics which may mitigate the risks associated with adopting the wrong cloud service provider.

5.5.1 Goal Tree of Indus

Figure 5.2 shows the goal tree of Indus cloud. Many obstacles were identified to achieve security while migrating to the Indus cloud. The goal tree of Indus cloud informs users of the potential risks that are involved with Indus cloud adoption.

5.5.2 Smart Bank Goal Tree

A cloud architecture evaluation method was proposed by Faniyi et al. in [62] . A case study of a Smart bank was presented where the bank decides to adopt cloud services to deliver a robust, scalable and on-demand service provision for the risk management aspect of its

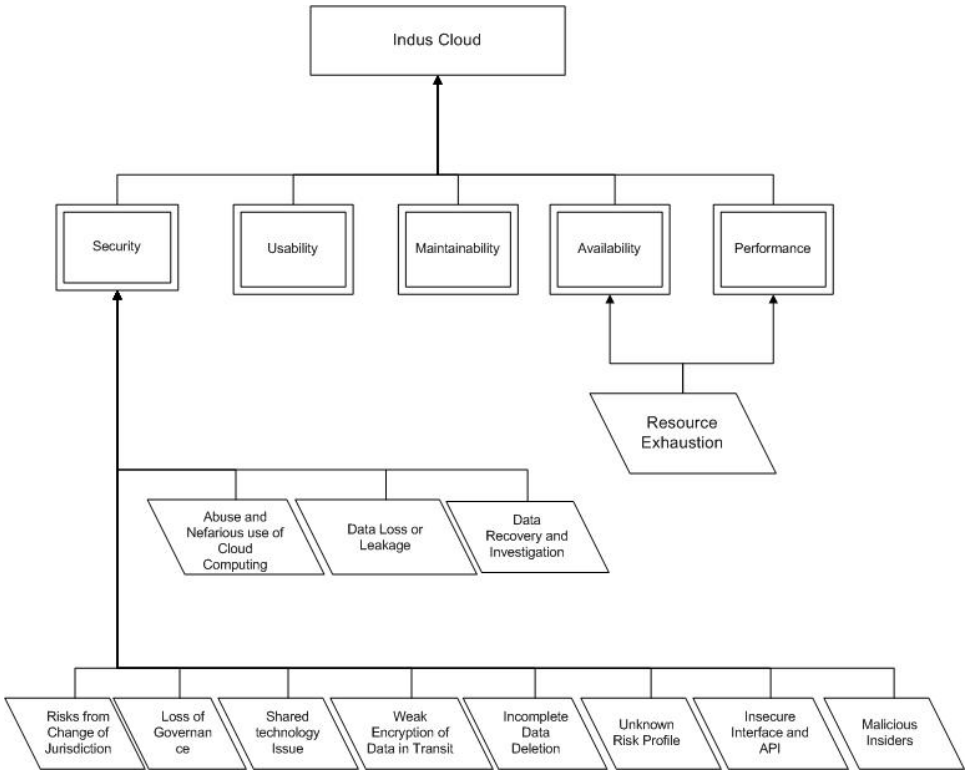

Figure 5.2: Goal tree of obstacles to achieve security in an Indus Cloud

business. The architecture proposed was evaluated using Architectural Tradeoffs Analysis Method (ATAM) [88] incorporating dynamic analysis via implied scenario generation and was named as ATMIS.

Goal 1

Goal Achieve [Store Sensitive Data in home country]

Category Security

Definition Store Sensitive data in the country of operation.

Smart Bank policies prohibit some of the sensitive data from being hosted on any server outside its country of operation. As cloud acts as a black box and the user has little or no knowledge about the location of the data centre; this poses a threat to Smart Bank's content. SLA clauses often make no mention of any sub-contracting of data centres. Data from Linode, a leading cloud service provider was unavailable to the users for about 20 minutes due to a power outage at Hurricane Electric which owns the data centre where Linode stores data

[2]. The outage at Hurricane Electric was completely outside the control of Linode. Such sub-contracting poses a grave threat to Smart Bank's policies.

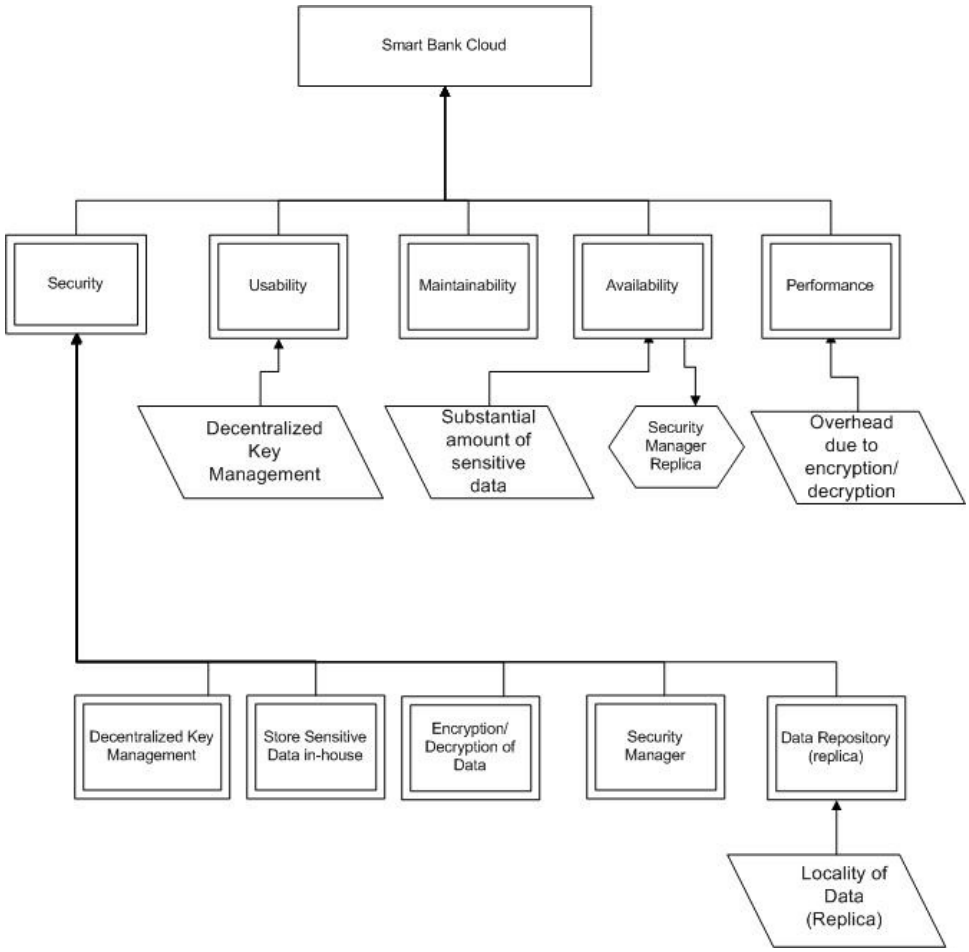

Figure 5.3: Portion of the goal tree for Smart Bank

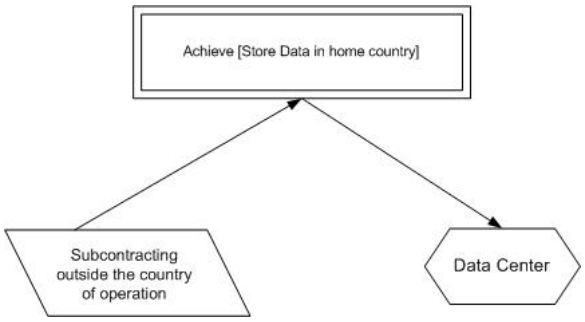

Figure 5.4: Obstacle for achieving Goal 1

5.6 Resolving Obstacles in the Process of Cloud Adoption

By looking at both the perspectives of the case study i.e. Indus (see section 5.5.1) and Smart Bank (see section 5.5.2) we will try to map the requirements of the user with the features of Indus. Features of the cloud service provider may be obtained from the cloud SLAs, market reviews, benchmarks, whitepapers etc. In the case of Indus we have looked at the features of Indus through the Indus Web Services Customer Agreement, SLAs, whitepapers and benchmarks. The cloud service provider may indicate all the available features in their SLA. However, other sources of information like benchmarks and market reviews can also be included to know more about the cloud features. We are proposing some concrete and abstract obstacle resolution tactics which could help the user in minimizing the risks associated with cloud adoption. Table 5.2 shows the obstacle resolution tactics. Some of the obstacles cannot be fully resolved. We can set the priority of resolving the obstacles by quantifying the value of the obstacles. Every obstacle has a value. The value of an obstacle is the product of the likelihood of the obstacle occurrence and the consequence of that obstacle. As obstacles lead to risks therefore they should be quantified in order to know the damage that may be caused due to the resulting risks.

$$V(O) = L(O) * C(O) \tag{5.1}$$

Where L(O) is the likelihood that obstacle O will occur, C(O) is the cost/consequence, and V(O) is the value of obstacle O. For simplicity we will rate each on a scale of 1-4 scales. The larger the number the larger would be the consequence or likelihood. Priority for resolving

the obstacle can be established by using the matrix shown in figure 5.5. Equation (1) is based on Boehm et al. equation for quantifying the risks in a software development process [29] . It may be possible that the obstacle resolution for a particular obstacle is more expensive than the value of the damage that may be caused due to the obstacle itself. Some obstacle may be resolved to a certain degree but may not be fully resolved. Table 5.2 defines some obstacle resolution tactics in the process of cloud adoption.

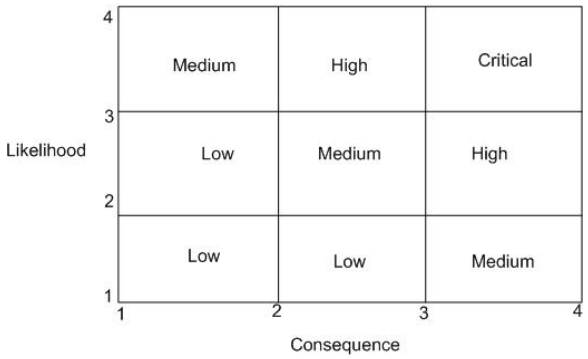

Figure 5.5: Matrix for quantifying the value of obstacles

5.6.1 Obstacle Prevention

The obstacle prevention strategy resolves the obstacle by adding a new goal. By adding a new goal obstacle occurrence can be avoided.

Tactic 1: The goal tree of Indus reveals an obstacle of resource exhaustion. The same obstacle was identified in the Smart Bank case study where the sensitive information is stored in-house. If the amount of sensitive information is significant than we have to move sensitive data to the cloud. Smart Bank can store the sensitive data in the cloud in encrypted form using Trusted Platform Module (TPM)[133]. Smart Bank therefore adds a new goal to prevent the obstacle. Using TPM is a new goal which has been added to the system for avoiding the obstacle. This will help Smart Bank to avoid the maintenance of data repository in-house. See Figure 5.6

Tactic 2: Many cloud providers subcontract part of their services. The issue of subcontracting cloud services is complicated. Who will have access to user data and where is the

subcontractor located are of paramount importance to the user. Subcontracting makes transparency difficult. Usually when there is a dispute between the cloud service provider and the user, the cloud service provider tries to shift the liability to the subcontractor and the customer may not have the right to take any action against the subcontractor [74]. Let's take Linode example where the data center has been subcontracted to Hurricane Electric. In this situation if Smart Bank wants to exploit the services of Linode then the SLA should clearly state that the data center is not owned by Linode. SLA should also mention the location of the data center as Smart Bank does not want its data to be stored outside its country of operation. This will allow the user to make decision by keeping their requirements satisfied. The cloud service provider should provide the list of all the subcontractors to the user. Also, if there is any change in the subcontractors then it should be first informed to the users. Negotiation of SLA is another goal which has been added to the goal tree to prevent any the obstacle shown in Figure 5.4. See Figure 5.7 for the resolution to this obstacle and Figure 5.8 for the goal tree of this tactic.

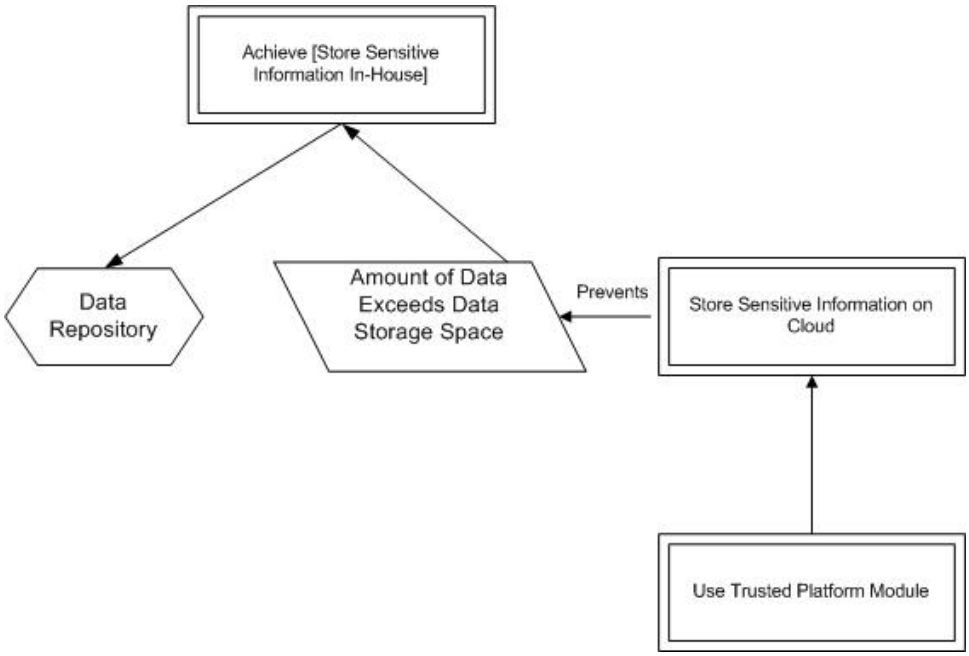

Figure 5.6: Obstacle Resolution Tactic1

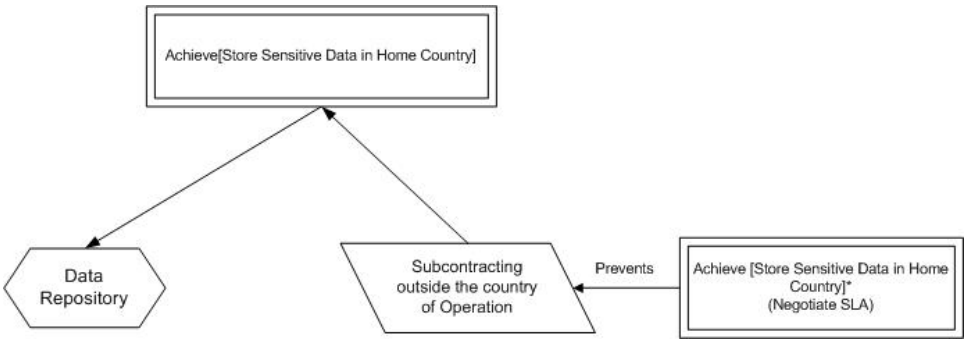

Figure 5.7: Obstacle Resolution Tactic2

5.6.2 Cloud Service Substitution

Tactic 3: Smart Bank wants its system to be available 24/7. We assume that Bank also wants to scale up when there is increased traffic and scale down when the demand is low. Nallur et al. have proposed a self-optimising architecture [120] where the website can scale up and scale down according to the traffic. Assume that when the traffic is low the website is using cloud C1 which has the capacity to deal with fewer requests and when the traffic is high the website starts using cloud C2 which can handle more requests. This means that the web service can scale up when we substitute the service with another one during the peak hours. This strategy resolves the obstacle of non-availability of the website when the traffic is high. Smart Bank can sign up to the services of different clouds for different time. This will make the website available at all times (see Figure 5.9). Hybrid clouds are one such example where different clouds are used by the same organization to handle the traffic during low and peak demands [110].

Goal 2

Goal Achieve [24/7 Availability of Smart Bank]

Category Availability

Definition The website should always be available to the users.

Tactic 4: This tactic consists of splitting a goal assigned into subgoals, so that every subgoal can be assigned to a different cloud to satisfy it. Application of this tactic may lead to the

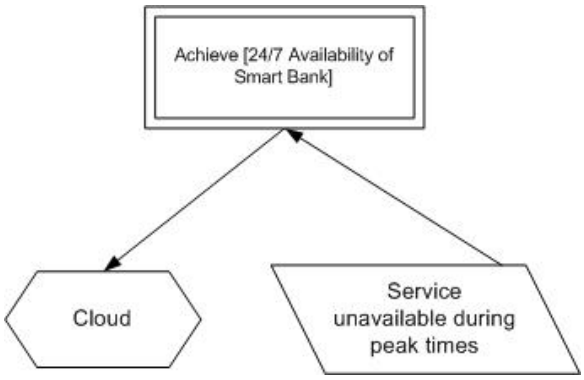

Figure 5.8: Goal Tree of Goal 2

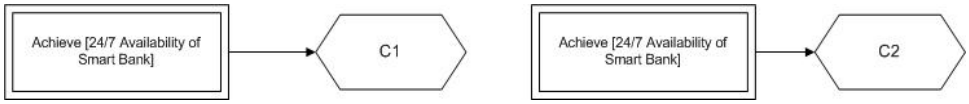

Figure 5.9: Obstacle Resolution Tactic 3

introduction of some new clouds to satisfy the subgoals. As web services hosted on the cloud evolve with the passage of time, their requirements evolve accordingly. Smart Bank is a new bank with fewer customers. With the passage of time the bank will become more popular and it will be critical for Smart Bank to be highly stable, secure and available. Downtime of Smart Bank is unacceptable to its customers. Such web applications which have become popular among the users need to adapt their architecture with the growing demand. Assume that the data centre D1 was used by Smart Bank but as the demand grew Smart Bank has to lease another data centre to avoid any chances of unavailability to the user. The SLA can be negotiated to lease another data centre from a cloud different than the current one so that data centres D1 and D2 are owned by two different cloud service providers. See Figure 5.10

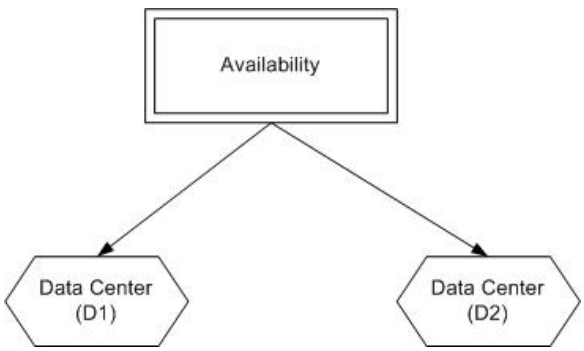

Figure 5.10: Obstacle Resolution Tactic 4

5.6.3 Goal Weakening

Tactic 5: This tactic involves weakening the goal definition or the degree of satisfaction. As an example UoB wants the response time for its email service to be 10 seconds but agrees to settle for 12 seconds if the Cloud Service Provider C1 is unable to guarantee a response time of 10 seconds. The satisfaction degree of the goal was relaxed to avoid the elimination of a potential Cloud Service Provider for service provision. See Figure 5.11

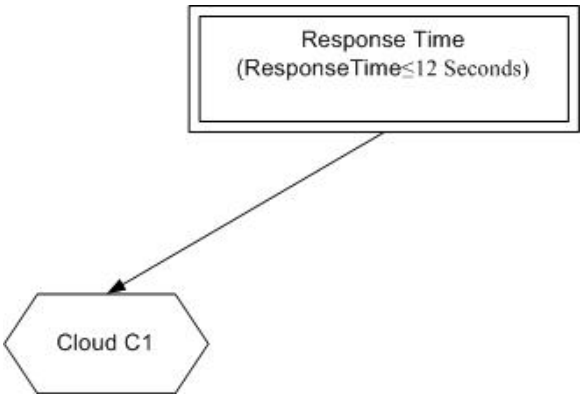

Figure 5.11: Obstacle Resolution Tactic 5

5.7 Goal Prioritization

Goal prioritization is a core activity for resolving the obstacles in the requirements engineering process for cloud adoption. Stakeholders engage in an extensive prioritization process to distinguish core goals (i.e. critical goals that should always be satisfied) from the desirable goals (i.e. goals that can be traded off). There might be some obstacles which could be avoided due to their significantly lower impact on the system. However, obstacles that can critically affect the system should be resolved with high priority. In the goal prioritization process goals are assigned weights according to the stakeholders' preferences. We describe an approach which helps in prioritizing the goals and obstacles. The highest level of goal refinement tree is level 0, the next level of subgoals is level 1 and so on. We assign the weights to goals in each level, so that the sum of the weights of goals in each level is 1.

Step:1 Obtain the relative importance of the goals

Consider the goal tree of smart bank. In level 1 we have five goals g_1, g_2, g_3, g_4, g_5 and now we want to know their relative value. Rank the importance of each goal. Smart Bank prefers security over everything else. After comparing and obtaining the relative importance of the level 1 goals, we have that:

$g_1 > g_4 \approx g_5 > g_3 > g_2$

This implies that g_1 is the most important goal followed by g_4 and g_5 (g_4 and g_5 are of equal importance), g_3, and g_2. The weight μ of each goal is:

$\mu g_1 > \mu g_4 \approx \mu g_5 > \mu g_3 > \mu g_2$

In a similar way stakeholders compare the relative importance of goals of the same parent in different levels.

Step 2: Assign each goal its weight based on the obtained relative importance

Stakeholders' engage in brain storming sessions to discuss how the satisfaction of each goal can contribute to the achievement of strategic objectives. These sessions lead to an agreement among stakeholders about their priorities and assigning weights to goals. We continue the process of assigning weights to the goals belonging to the same parent. For level 0 goals (see Figure 5.12) we assign 0.3 to g_1 as security was deemed most important by the stakeholders. Since g_4 and g_5 have an equal importance therefore they have been assigned the same weight. The weights have been assigned to the goals keeping in view their relative importance.

Step 3: Obtain composed weights for offspring goals

The final weight of the goal is called a composed weight. Composed weight is the product of the weight of each goal with the weights of all the parent goals. We can obtain the composed

weight for the obstacles $\neg g_2$, $\neg g_4$, $\neg g_5$ and $\neg g_{1.5}$, so that:

$\mu g_2 = 0.1$

$\mu\ g_0 = 1$

$\omega g_2 = \mu g_2 \text{ x } \mu g_0$

$\omega g_2 = 0.1 \text{ x } 1 = 0.1$

Similarly the composed weights for other obstacles can be calculated

$\omega \neg g_4 = 0.25$

$\omega \neg g_5 = 0.25$

$\omega \neg g_{1.5} = 0.051$

This technique of prioritizing goals by comparing the offspring of the same parent allow decision makers to reason about the importance of a particular goal in comparison to other goals.

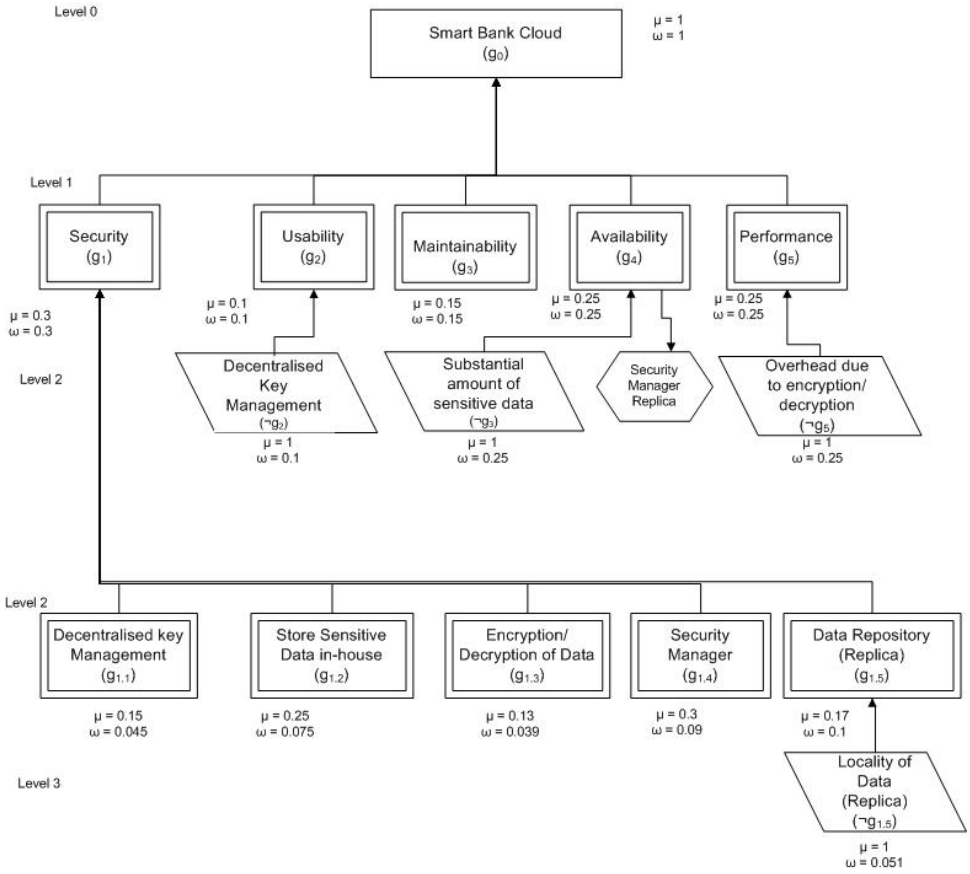

Figure 5.12: Obstacle Resolution Tactic 5

5.8 Discussion & Conclusion

We identified several risks associated with cloud adoption after looking at the SLAs and white-papers of different clouds. We have introduced the notion of obstacles to the cloud adoption lifecycle defined earlier in chapter 4. Our approach is helpful in identifying the conflicts between requirements, tradeoffs and risks. We likened obstacles to risks and proposed some concrete and abstract obstacle resolution tactics leveraging on the work of [162]. To our knowledge no work has modelled the obstacles for cloud adoption process. [9] has defined a method for goal prioritization. The method simply prioritize goals without taking obstacles into consideration. We prioritize obstacles using the same method. We have also defined a framework for prioritizing user requirements and obstacles. The framework is based on utility theory. Prioritizing the requirements helps users to resolve the most critical obstacles first and to achieve goals that are most important. Prior to our work to the best of our knowledge

no one has prioritized obstacles.

We have defined the steps involved in the requirements engineering phase for cloud adoption. This chapter contributes a lifecycle which is an extension of our previous work presented in chapter 4. The lifecycle could help in mitigating the risks which may arise due to incomplete requirements or insufficient cloud features. We have presented a systematic guidance for the identification of the obstacles and have proposed some concrete and abstract obstacle resolution tactics to minimize the risks associated with cloud adoption. We have also used utility theory to prioritized the goals for minimizing the risks.

CHAPTER 6

Cloud Adoption: Prioritizing Obstacles and Obstacles Resolution Tactics Using Analytical Hierarchy Process

Things which matter most must never be at the mercy of things which matter least.

Johann Wolfgang von Goethe

6.1 Introduction

In chapter 4, we have contributed to a novel Goal-Oriented Requirements Engineering method for cloud adoption. In chapter 5 we refined the method and introduced the notion of obstacles. We also proposed some concrete and abstract obstacle resolution tactics in the previous chapters. We argue that obstacles prioritization and their resolution are important for mitigating risks in the adoption process. The novel contribution of this chapter is a new systematic method for prioritizing cloud adoption obstacles and their resolution tactics using Analytical Hierarchy Process (AHP). We demonstrate the applicability and usefulness of the approach through an example. We do the sensitivity and stability analysis to assess the AHP choice of the resolution tactics.

6.2 Why do we need prioritization?

Decision making becomes difficult when we have a large set of requirements, numerous risks, multiple stakeholders and many operationalization/design alternatives. To minimize the decision space it's inevitable to prioritize requirements and/or different alternatives. Prioritization should strive to meet the requirements subject to constraints like time, resources, expertise etc. [86]. According to [108] many organizations believe that it is important to prioritize requirements to make decisions about them based on rational, quantitative data. In a cloud case, requirements can be large due to the multi-tenant nature of the cloud and the number of stakeholders involved. Though the diversity of stakeholder background and input can be deemed to be a strength in RE for the cloud, it is expected that conflicts and inconsistencies are the norm. Prioritization of requirements will reduce the space of requirements that needs to be negotiated against the cloud service provisions. The requirements can be conflicting and therefore it is expected to negotiate the requirements against cloud service provisions. Once the requirements are prioritized, the user can decide which requirements can be negotiated and which of the requirements cannot. The choice of the requirements is often driven by their importance, criticality to the application, their ability to unlock application

potentials, and ability of the system to comply to standards.

In the context of our solution for the problem of matching cloud features to user requirements, it's key to make the matching process simple, effective and scalable. Mismatches can be large and can be indicative of risks. Since the criticality of the mismatches can be significant for some important requirements, prioritization can introduce discipline and better scope our effort in the requirements which are deemed to be the most important. Not all risks may be critical and in some situations its better to accept the risk [167] due to its lower significance. To know the criticality of risks in any system it is better to prioritize them. We use AHP for prioritizing the risks and risk mitigation tactics in this chapter. As AHP has proven to be effective in prioritizing risks in several engineering domains such as [66] [80], we use AHP as the basis for prioritizing risks.

6.3 The Analytical Hierarchy Process for Handling Obstacles in Cloud Adoption Using An Example

Since its early applications in the seventies, the Analytical Hierarchy Process has found a wide interest both between decision theorists and appliers [130] [131]. The analytical hierarchy process is a decision-making method and is well described in [131]. An AHP hierarchy is a vehicle for structuring a decision problem. The classical AHP hierarchy consists of an overall goal, a group of options or alternatives for achieving the goal, and a set of criteria, which relate the alternatives to the goal. The criteria can be further decomposed into subcriteria, sub-subcriteria, and so forth. In a nutshell, the AHP hierarchy can then be analyzed through a sequence of pairwise comparisons using numerical scales of measurement; these could be subjective or objective metrics. The criteria are pairwise compared against the goal for importance. The alternatives are pairwise compared against each of the criteria for preference. The comparisons are computed and priorities are derived for each node in the hierarchy. The prioritization can then provide insights for the decision makers and analysts, which can inform the selection decision. The process is structured and systematic; it overcomes the

limitations of ad hoc decision making in complex problems with various alternatives and criteria. AHP is well described in many text books on the subject e.g. see [131]

6.3.1 Motivation for Obstacle Resolution in Cloud Adoption

AHP is intuitively appealing as an approach to the obstacle resolution problem. Its hierarchy can provide intuitive and simple ready means for structuring, tracing and analyzing goals, obstacles, and their resolution tactics. Complementing the AHP structure with its analysis and prioritization process, the combination does provide a new systematic method for prioritizing obstacles and selecting suitable tactics for resolving them for the goals to be achieved. In this context, an obstacle resolution tactic corresponds to an alternative. Resolution tactics are pairwise compared against each of the identified obstacles for preference to the extent to which its resolution can satisfy the goal. Obstacles correspond to criteria, which can be further decomposed and refined into sub-obstacles; it's negation has relative importance for reaching the goal (these are pairwise compared against the goal for importance).

There is a general lack of systematic methods for prioritizing obstacles and identifying the best strategies for mitigating the risks. We systematically model requirements for cloud adoption and their corresponding obstacles. Obstacles can be deemed as risks; they can have a significant impact on the adoption process if not properly elicited, modeled and mitigated. We propose a two-phase risk aware method for cloud adoption benefiting from Analytical Hierarchy Process. The first phase elicits and prioritizes obstacles by understanding their potential impact on adoption. The second phase searches for good-enough resolution tactics for the prioritized obstacles so they can be mitigated for risks. Stability and sensitivity analysis is proposed to assess the choice of the resolution tactics.

We argue that there are significant benefits to using AHP to manage the prioritization of obstacles and their resolution tactics. Firstly, it is beneficial to quantify obstacles' importance through understanding their consequences on the adoption process, if not properly managed and mitigated for risks. Secondly, the adoption process is often constrained by limited resources (e.g., budget, man-months, schedule), exhaustive treatments for all obstacles is often

resource demanding and may be difficult. A “selective" strategy, through obstacles prioritization is a sensible and wise option to tackle the complexity of the space and to focus efforts on the critical obstacles, which cannot wait. The strategy can elevate the consideration of those, which are likely to be critical and downgrade and eliminate others, which can be tolerated or have little/no risk impact. As obstacles can be resolved using several alternative resolution tactics, each may provide different added value. Thirdly, the importance of obstacles can vary by the orders of magnitude. Quantification of their relative importance and significance for risks can provide an objective assessment for their management and resolution. In contrast to the criticism about AHP's subjectivity [86], we argue that AHP pair-wise comparison technique can be a cost-effective strategy to probe for better testing and understanding of the presence and the importance of obstacles. The pair-wise comparison can be risk revealing. The pair-wise comparison comprises much redundancy and is therefore less sensitive to judgmental errors. AHP points to inconsistencies by calculating a consistency ratio of judgmental errors. The smaller the consistency ratio the lesser the inconsistency, hence the more reliable results. Fourthly, we acknowledge that it may be problematic to scale up for larger projects using AHP. Karlsson et al. [86] recommend grouping the requirements, where grouping requirements can reduce the number of comparisons in an efficient manner. We believe that our use of GORE for cloud adoption and our handling of obstacles using GORE do provide an elegant grouping benefiting from the goal refinements hierarchy, promoting potential scalability. Fifthly, Karlsson et al. [86] evaluated six different prioritization techniques based on pair-wise comparisons and concluded that AHP was the most promising approach because it is based on a ratio scale and includes a consistency check. Last but not least, the major disadvantage of AHP is that it is time consuming for large problems. Research efforts have been made to decrease the number of comparisons and hence the time needed [85][112]. The result has been that the number of comparisons can be decreased by as much as 75% [85].

Guidelines for mitigating risks through resolving obstacles

1. **Define risk mitigation/obstacle resolution tactics**

   Some abstract and concrete obstacle resolution tactics have been defined in this thesis. Some risks may be accepted due to their non-criticality.

2. **AHP hierarchy**

   Make an AHP hierarchy of the identified obstacles and the alternatives (Obstacle resolution tactics). The AHP hierarchy typically has three levels: goal, criteria and alternatives. The goals can be different for different problems. Our goal is to mitigate the risk through obstacle resolution, criteria are different obstacles and alternatives are the obstacle resolution tactics.

3. **Prioritize obstacles**

   Prioritize the obstacles according to their criticality in the system by pair-wise comparisons using AHP

4. **Prioritize tactics**

   Prioritize the obstacle resolution tactics for every obstacle that is to know which obstacle resolution tactic is best suited for resolving a particular obstacle

5. **Stability Analysis**

   Perform the stability analysis for the prioritized obstacle and obstacle resolution tactics.

6. **Resolve obstacle**

   If the solution is stable (i.e. if the solution does not change with small perturbations in the values of the alternatives), resolve the obstacle.

   Note: Different stakeholders may have different priorities. Aggregation of the priorities assigned by different stakeholders may give us the overall ranking [4].

6.3.2 Obstacles and Risks in Requirements Engineering for Cloud Adoption

In [179], the authors have demonstrated that by analyzing Service Level Agreements (SLAs) of cloud providers and matching them against users' requirements in the goal refinement process, potential SLA violations, conflicts and likely risks can be revealed. In the context of GORE, the potential violations of the SLAs can be best modeled, analyzed, linked to consumers' goals and mitigated for risks using obstacle analysis [179].

Using obstacle analysis to mitigate likely risks of cloud SLAs violations, start by examining the consumers' goals, where obstacles are identified from the goal graph and terminal goals, which need to be assigned to cloud agents. Cloud agents can be a software-as-service, platform-as-service and/or infrastructure-as-service. The obstacles can be recursively refined; it is essential that the set of identified obstacles is complete for every goal (i.e. at least for high-priority goals). To resolve obstacles we would like to identify as many obstacles as possible for the goals, particularly for high-priority goals. A set of obstacles O1, ... ,On is complete for goal G if the following condition is true: ¬O1 , . . .,¬On|= G, which means that if none of the identified obstacles occur than the goal is satisfied [163]. In [179], the interlink between risks and obstacles for the cloud adoption problem had been systematically modeled. A sample of obstacle resolution tactics is depicted in Table 5.2. The user can select among these tactics to resolve the identified obstacles and mitigate risks in the adoption process. Refer to chapter 5 for details.

Figure 6.1 describes the high-level steps involved in resolving the obstacles in the cloud adoption process. Given the likely large space of possible obstacles and their resolution tactics, we use AHP for their prioritization. In the first phase obstacles are prioritized based on their relative importance and significance to the goal satisfaction. In the second phase, the application of AHP aims at informing the decision of reaching the optimal set of tactic(s) for resolving obstacles given numerous candidate alternatives. This is done by understanding their relative importance and significance upon resolving the tactics. A distinctive feature of our method is the use of sensitivity analysis to assess the stability of the solution before we

decide on a tactic to resolve the obstacle. As the assignment of weights and alternative values with respect to subjective criteria are approximate, it is preferable to get a stable solution. In other words, the order of the solution should not change with small perturbations in the values of the alternatives. If the solution is stable, we will proceed to resolve the obstacles. The method and its working steps are sufficiently described in section 6.4.

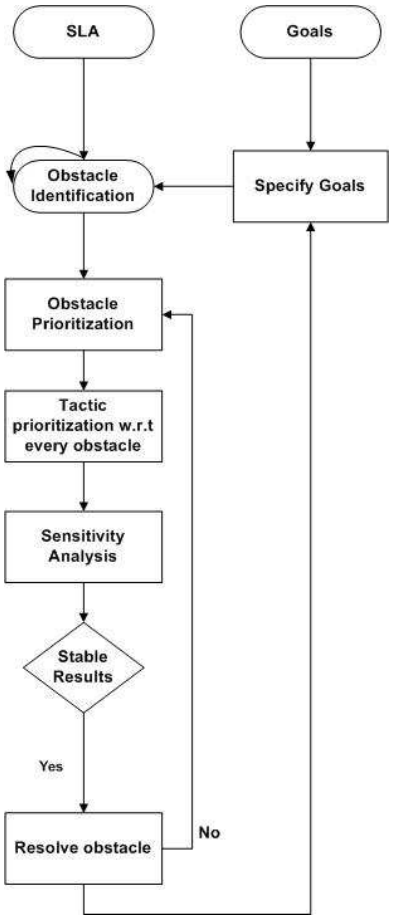

Figure 6.1: Steps for resolving obstacles

6.4 The Process of Obstacle Resolution for Cloud Adoption through an Example

We use an Industrial case study reported in [62] by Faniyi et al. referred to as Smart Bank as an example to show how AHP can be applied for mitigating risks in the process of cloud adoption. Smart Bank was interested in assessing the architectural design decisions for

security upon moving to the cloud. In [62], the architecture design decisions were evaluated using Architectural Tradeoffs Analysis Method (ATAM) incorporating dynamic analysis via implied scenario generation and were named as ATMIS. Looking at the case, we have identified the goals and the obstacles that may be encountered for achieving those goals using GORE, as described in Figure 5.3.

6.4.1 Building a Heirarchy for Smart Bank Obstacles

We use AHP for prioritizing the obstacles and obstacle resolution tactics. As the decision in the obstacles resolution process revolved around goals, their likely obstacles and their candidate tactics for resolving them; it is perfectly justified to model the obstacles resolution process using AHP primitives. Figure 6.2 shows the hierarchy of goals, obstacles and the obstacle resolution tactics. The overall goal in our case is to resolve obstacles in the smart bank example. We build the hierarchy from the top (depicting goals) through the intermediate levels (depicting obstacles) and moving to the lowest level, which comprises the list of alternative resolution tactics. The major decision criteria occupy the second level of the hierarchy (i.e. which obstacles should be resolved and their relative importance for the goal's satisfaction), and the sub-criteria occupy the third level of the hierarchy. The last level of the hierarchy has all the alternative options. Four obstacles were identified for achieving different goals of the smart bank. For simplicity we are only making a hierarchy of the main goal, their obstacles and resolution tactics. We have not included the subgoals in the hierarchy for the simplicity of exposition.

6.4.2 Scale for Pairwise Comparisons

Prioritization of both obstacles and obstacle resolution tactics is done through pair-wise comparison. Thomas L. Saaty [131] proposed a 9-grade value scale to compare different choices. Scale ranges from 1 (equal value) to 9 (where an option is extremely more valuable than the other). In the first phase, obstacles are prioritized based on their relative importance

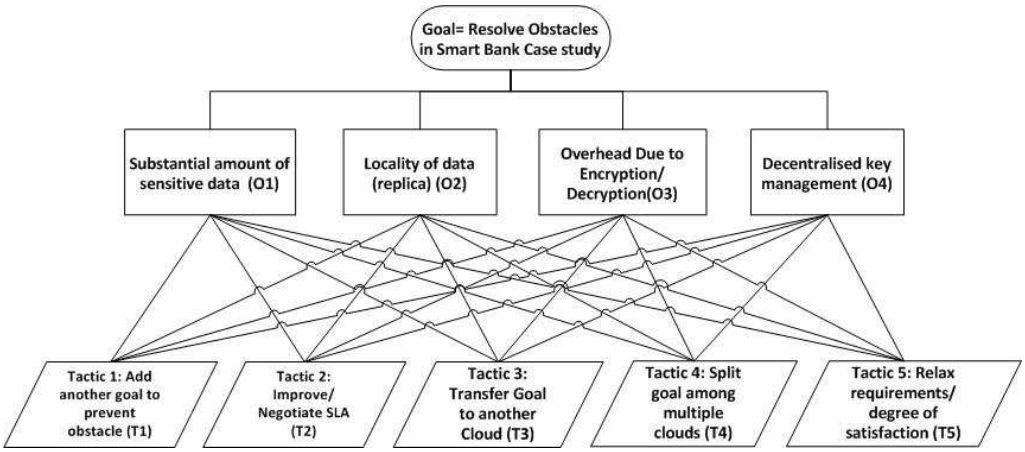

Figure 6.2: Hierarchy for Smart Bank

and significance to the goal satisfaction. In the second phase, the application of AHP aims at informing the decision of reaching the optimal set of tactic(s) for resolving obstacles given numerous alternative candidate. Understanding their relative importance and significance upon resolving the tactics does this. By using AHP the user can prioritize the obstacles that should be resolved first and can select the best obstacle resolution tactic for a particular obstacle. We use Table 6.1 for valuing the pairwise comparisons. We refer obstacle Substantial Amount of sensitive data as O1, obstacle Locality of data as O2, obstacle Overhead due to encryption/decryption as O3 and obstacle Decentralized key management as O4.

6.4.3 Prioritize Obstacles Using AHP

Prioritize the obstacles by pairwise comparisons using Table 6.1. The values are inserted in a matrix. Since there are 4 obstacles, the matrix is 4x4. An element is equally important when compared to itself, so where the row of O1 and column of O1 meet in position (O1, O1) insert 1. The diagonal of the matrix therefore must consist of 1's. For each pair of obstacles (starting with O1and O2, for example) insert their determined relative priority in the position (O1, O2) where the row of O1 meets the column of O2. In position (O2, O1) insert the reciprocal value. Continue to perform pairwise comparisons of O1-O3, O1-O4, and so on.

Thus for prioritizing obstacles, six pairwise comparisons are required. The values that are inserted in the matrix can be obtained by a voting procedure where each stakeholder is

Table 6.1: Scale for pairwise comparisons

Relative Importance	Short Definition	Explanation
1	Equally important	Two obstacles are equally critical
3	Weakly more important	Slightly favour resolving one obstacle over another
5	Strongly more important	Strongly favour obstacle for resolving over another
7	Very strongly more important	An obstacle resolution is strongly favoured and its dominance is demonstrated in practice
9	Extremely more important	The evidence favouring one over another is of the highest possible order of affirmation
2, 4, 6, 8	Intermediate values between two adjacent judgments	Used to facilitate compromise between two slightly differing judgments
Reciprocals	If obstacle A has one of the above numbers assigned to it when compared with obstacle B, then B has a reciprocal value when compared with A.	

allowed to allocate a number from Table 6.1 to the obstacles. Stakeholders who should be involved in the prioritization of the obstacles can be identified using the already existing tools like StakeNet [105].

Pairwise comparison of obstacles according to their criticallity on the system gives the following matrix

$$\begin{array}{c} \\ O1 \\ O2 \\ O3 \\ O4 \end{array} \begin{array}{c} \begin{array}{cccc} O1 & O2 & O3 & O4 \end{array} \\ \begin{pmatrix} 1 & 5 & 3 & 7 \\ 0.20 & 1 & 0.33 & 5 \\ 0.33 & 3 & 1 & 5 \\ 0.14 & 0.20 & 0.20 & 1 \end{pmatrix} \end{array}$$

After comparisons, normalize each column (add its components and divide each component by the sum of that column). We obtain the following matrix.

$$\begin{pmatrix} 0.5966 & 0.5435 & 0.6618 & 0.3889 \\ 0.1193 & 0.1087 & 0.0735 & 0.2778 \\ 0.1989 & 0.3261 & 0.2206 & 0.2778 \\ 0.0852 & 0.0217 & 0.0441 & 0.0556 \end{pmatrix}$$

The sum of the rows of the normalized matrix is a column vector which when averaged by the size of 4 columns yields the vector of priorities. After obtaining the priority vector, multiply the vector with matrix of comparisons, which will result in a new product vector. Divide the first component of the product vector with the first component of priority vector, the second component of the product vector with second component of the priority vector and so on, we obtain another vector called the ratio vector. The calculations made for prioritizing the obstacles reveal that O1 has the highest priority followed by O3, O2 and O4 respectively.

$$\begin{pmatrix} Priority & Product & Ratio \\ 0.5477 & 2.400939 & 4.383829 \\ 0.1448 & 0.597942 & 4.128573 \\ 0.2558 & 1.13118 & 4.421621 \\ 0.0517 & 0.210032 & 4.065666 \end{pmatrix}$$

Take the sum of the components of the ratio vector and divide it by the number of components we have an approximation to a number λ (called the principal eigenvalue). The closer the λ is to n (the number of activities in a matrix) the more consistent is the result. The deviation from consistency is represented by (λ-n)/ (n-1) which is the consistency index (CI). The ratio of consistency index to random index (RI) is called the consistency ratio (CR). The values of random index for matrix of different orders are defined in [131]. The consistency ratio of 0.10 or less is considered acceptable. For obstacles the consistency ratio is acceptable.

CI= 0.083307

CI/RI= 0.09

6.4.4 Selecting an Appropriate Obstacle Resolution Tactic

We compare all the five obstacle resolution tactics with respect to every obstacle for selecting the best resolution tactic. We calculated the priority of obstacle resolution tactics for the four major obstacles using AHP. Section 6.5 shows the results of calculations in the form of a matrix.

For decentralized key management the best tactic is Tactic 5. For the obstacle overhead due to encryption/decryption the best tactic is Tactic 4 where the bank can keep its data on two different providers' servers. Obstacle substantial amount of sensitive data can be resolved using Tactic 1 which has the highest priority. For the Locality of data the best obstacle resolution tactic is Tactic 4 i.e. smart bank should engage into negotiation before signing up the contract for cloud to make clear where the data of the bank will be stored.

6.5 Stability and Sensitivity Analysis

The selection of obstacle resolution tactics can be considered as a multicriteria decision problem. When solving a multicriteria decision problem, it is desirable to choose a decision function that leads to as stable a solution as possible. Since the assignment of weights to the tactics and obstacles is based on human judgment, it would be preferable to obtain the same solution (i.e. to have a stable solution) when slightly modifying the weights. We consider the decision functions based on α power means [59]:

$$\left(\frac{\sum_{i=1}^{n} w_i {a_i}^{\alpha}}{\sum_{i=1}^{n} w_i}\right)^{1/\alpha} \tag{6.1}$$

Where the weights w_i in our case are the priority vector for the obstacles and the alternative values a_i are the different tactics.

We calculate the stability index of our multicriteria decision functions for perturbations of alternative values ϵ as:

$$S(\epsilon) = (\Pi_{j=1}^{n-1} \delta_j(\epsilon))^{\frac{1}{n-1}} \tag{6.2}$$

where δ_j is the stability value of the order between alternative j and j+1, if the alternative

values are allowed to vary by at most ϵ. A value of S(ϵ)=1 is equivalent to a stable solution whereas a value of S(ϵ)= 0 corresponds to an unstable solution. In order to calculate the stability of our solution, we put the priority vectors in a matrix for every obstacle:

$$\begin{array}{c} \\ O1 \\ O2 \\ O3 \\ O4 \end{array} \begin{array}{c} \begin{array}{cccccc} Weight & T1 & T2 & T3 & T4 & T5 \end{array} \\ \begin{pmatrix} 0.5477 & 0.5083 & 0.2306 & 0.1357 & 0.0735 & 0.0519 \\ 0.1448 & 0.0459 & 0.5095 & 0.2197 & 0.1418 & 0.0832 \\ 0.2558 & 0.0439 & 0.0749 & 0.1573 & 0.5186 & 0.2052 \\ 0.0517 & 0.2261 & 0.1316 & 0.0782 & 0.0428 & 0.5213 \end{pmatrix} \end{array}$$

The α power mean aggregated values for the different tactics for values of α of 2, 3 and 5 are shown in Table 6.2.

Table 6.2: α power mean aggregated values for different tactics

α	T1	T2	T3	T4	T5
2	0.3807	0.2627	0.1540	0.2734	0.1652
3	0.4170	0.2966	0.1577	0.3311	0.2132
5	0.4507	0.3510	0.1654	0.3949	0.2909

We consider two situations: potential changes of up to ϵ=100% of the alternative values and potential changes of up to ϵ=10% of the alternative values. We calculate the stability values for the order for each pair of consecutive alternatives $\delta_j(\epsilon)$ and then the stability index S(ϵ). Table 6.3 shows the calculated stability index.

The stability index in both cases is highest for α=3, suggesting that an overall solution this is the best. For variations of up to 10% of the original value the 3-power mean decision function is close to perfect stability. However, the use of the 2-power mean is recommended if only the order of the first two tactics is of interest, given that the best tactic will be selected and the largest δ value for the order between tactic T1 and T4 is achieved when using the 2-power mean.

Table 6.3: Stability index for resolution tactics

α	Stability Index S(0.1)	Stability Index S(1)	δ(1) \ Priority(tactic) 1 (T1)	2 (T4)	3 (T2)	4 (T5)
2	0.4886	0.0669	0.16	0.019	0.22	0.03
3	0.8612	0.1079	0.11	0.055	0.16	0.14
5	0.7632	0.0978	0.065	0.058	0.09	0.27

6.6 Conclusions

This chapter presents a systematic risk-aware method for prioritizing obstacles encountered in the cloud adoption process and their resolution tactics using Analytical Hierarchy Process. The method leverages on goal-oriented requirements engineering, obstacles handling and AHP to systematically model risks and their mitigation strategies. We have exemplified the steps of the method using a non-trivial case to demonstrate its applicability. To evaluate the accuracy of the approach, we have used stability and sensitivity analysis. The analysis aims to assess the AHP choice of resolution tactics, which have the potentials to mitigate risks. We have observed that for variations of up to 10% of the original value, the 3-power mean decision function tend to be close to perfect stability. This means that the method can accommodate up to 10% “discrepancy" upon the allocation of weight by various stakeholders.

CHAPTER 7

Evaluation

> Fear cannot be banished, but it can be calm and without panic; it can be mitigated by reason and evaluation.
>
> *Vannevar Bush*

In the previous chapters we have defined a method which will guide the screening and selection of cloud services. The key objective of this method is to support the analysis of mismatches and the resolution of mismatches that occur during the evaluation of the cloud services between user requirements and cloud service providers.

The primary objective of our method is to select an optimal cloud according to user requirements and mitigating the risks associated with cloud adoption. In this chapter we evaluate the feasibility and usefulness of our method. The nature of the decision made for adopting and selecting the cloud service varies case-by-case.

The chosen strategy to evaluate our method is through case studies. Case studies have been extensively used to empirically assess software engineering approaches [109]. Case studies provide an empirical and practical evidence for showing the suitability of a method for solving a particular class of problems. Since case studies cannot be reproduced, it is difficult to generate meaningful results that can be generalized [170]. In order to control the results of the evaluation effort, we conducted the case studies according to the DESMET methodology [91] for guiding the evaluation of software engineering methods. The authors state that while undertaking the case study, the first decision should be to determine what the case study aims to investigate and verify. In other words, before applying a method to a case study, define the goals. For evaluating our method, the goal we want to achieve is to show the validity of the following hypothesis:

"*The proposed method is a systematic way to identify the requirements in cloud service adoption and to mitigate the risks*"

In order to see the validity of the above mentioned hypothesis, it is important to define the effects we expect the method to have. This chapter describes the two case studies completed during the course of this PhD research to answer the following questions.

1. Why is goal-oriented a suitable requirements engineering method for specifying user requirements for cloud adoption?
2. How are the obstacle identification and obstacle resolution techniques useful for mitigating the risks associated with cloud adoption?

3. How can the method be useful for identifying the conflicts and tradeoffs in user requirements?
4. How we can determine the criticality of the risks through prioritization? Why is the choice of AHP for prioritization justified?
5. Does the priority of obstacles and obstacle resolution tactics remain the same after small perturbations in values of the alternatives?

DESMET has identified nine different evaluation methods [91]. We are using the *"qualitative screening"* method of evaluation which is a feature-based evaluation done by a single individual who not only decides the features to be assessed and their rating scale but also does the assessment [91]. We will also evaluate each phase of the method under the criteria of

1. **Completeness:** We are interested to see whether different phases of the method ensure completeness. Whether the acquired requirements and the identified risks are complete. We would like to see how GORE is helpful in achieving requirement completeness for cloud adoption. We would also like to evaluate how AHP ensures completeness in the prioritization process.
2. **Subjectivity:** The matching of cloud features with user requirements is subjective due to the involvement of different stakeholders. Different stakeholders may have different requirements. There may be conflicts between user requirements due to the tensions between different stakeholders [104]. As the weights for prioritization are assigned by the humans based on their judgment, subjectivity becomes an issue during the prioritization process. We would like to evaluate how subjectivity affects different phases of the method for cloud adoption.
3. **Scalability:** Since the cloud is a dynamic environment, there may be an increase in user requirements, stakeholders and risks. The evaluation will take into account the scalability of the proposed method. We would like to see whether the proposed method is able to scale for large number of requirements, stakeholders and risks.

   We looked at different case studies to evaluate our work. For example, the case study of an SME presented in [89]. The case study did not have much details on the goals

and hence was not thought to be suitable for evaluation. Another case study for cloud migration was presented in [149]. We had to drop this case study too as it wanted to migrate the .Net application to the cloud. There were no high level goals defined which could have been refined. A case study of Smart Bank has been earlier presented where we mitigated the security related risks in the cloud adoption process. We will now evaluate our framework using two different real world case studies. The first case study is about an SME wanting to adopt SaaS services for its Customer Relationship Management. The risk we have mitigated in this case study are mostly related to data security and identity management. The second case study is about "myki" which is a smartcard used as a ticket to travel on public transport in Victoria state of Australia. The risks in this case study are mostly related to scalability. We have therefore selected these two case studies to see how our proposed framework can help mitigate different kind of risks while adopting different types of cloud services.

7.1 Case Study 1

7.1.1 The case company

The SaaS solution is thought to be one of the effective ways for increasing the IT competitiveness of an organization due to many benefits which include speedy deployment, lower initial costs and no need for software maintenance. In Taiwan, SaaS is still at its earliest stages. Most of the small and medium enterprises are not yet using the SaaS solutions [168]. There seems to be a trust deficit between the vendors and the users. It is important to improve the trust between user and the cloud service provider by mitigating any perceived risks while cloud adoption. The case study has been reported in [168]. The case company F is an outstanding SME in Taiwan. The case company was established in 1969 and has distributors world over. It is one of the world's leading manufacturers in the niche and specialized resistor markets (especially focusing on Germany and Japan). Company F is wishing to serve as an advanced resistor technology platform with emphases on high flexibility and automation, high quality,

customization, professional R&D service team, one-stop shopping, just-in-time delivery system, and application consulting.

The company wants to introduce SaaS services from Salesforce.com for Customer Relationship Management (CRM). CRM enables businesses to focus on their relationships with different stakeholders whether they are users, suppliers, customers etc. The best thing about cloud based CRM system is that users can work from anywhere without installing the software on their machines. The company F formed a task force of five key people, including the General Manager and four managers from marketing, production, financial and information technology departments. After several group meetings the task force members decided to consider seven risks from [141] which included R_1 (data locality and security), R_2 (network and web application security), R_3 (data integrity and segregation), R_4 (authentication and authorization), R_5 (virtualization vulnerability), R_6 (data access and backup) and R_7 (identity management and sign on process). The reason behind the selection of this case study is to show the effectiveness of our method in mitigating the risks while adopting the cloud services. We will see how the proposed method helps in achieving the goals while minimizing the risks.

7.1.2 Goal Tree of Company F

The following figure 7.1 shows representative goals that need to be satisfied while adopting the CRM system. For simplicity of exposition we restrict our attention to three goals. These are access control and identity management, data security and integrity and software and system security. Nevertheless, the list of possible goals could be endless. These are the kind of priority goals which need to be screened. These goals are critical and will determine halting or continuation of the adoption process. The lack of satisfaction of these goals may determine whether we should adopt the cloud or not. If we need to go ahead then what are the possible risks and resolution tactics.

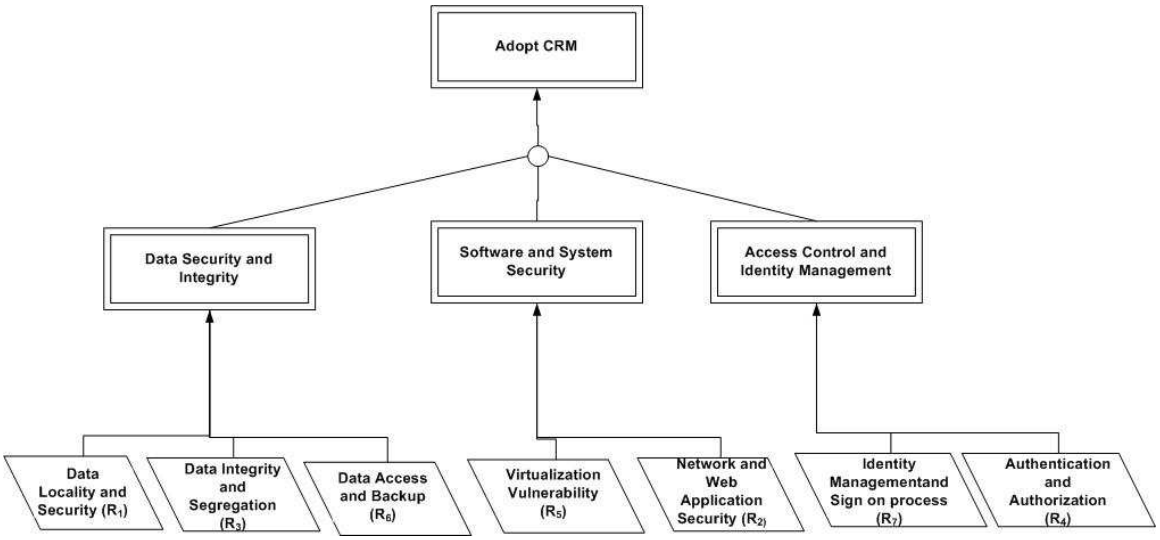

Figure 7.1: Goal tree of Company F

7.1.3 Building a Hierarchy for Company F Risks

We liken the risks to obstacles. We build a hierarchy starting from the goals (top level) through the risks (intermediate level) and moving to the risk mitigation tactics at the lowest level. The main decision criteria are at the second level of the hierarchy (i.e. what risks are to be mitigated and their relative importance for goal satisfaction). The third level of the hierarchy has all the alternatives which in our case are obstacle resolution/ risk mitigation tactics. Figure 7.2 shows the AHP hierarchy for mitigating Company F risks.

7.1.4 Prioritize Risks Using AHP

Prioritize the risks by pairwise comparisons using Table 6.1. The whole process of assigning weights is human centered and is significantly influenced by the perspective of the stakeholders. The stakeholders involved in prioritization will have to be identified as discussed in table 7.5. We have given the priorities to the risks according to their criticality in the system as mentioned in [168]

Since there are seven risks, the matrix is order 7x7. An element is equally important when compared to itself, so where the row of R_1 and column of R_1 meet in position (R_1, R_1) insert 1. The diagonal of the matrix therefore must consist of 1's. For each pair of risks (starting with R_1

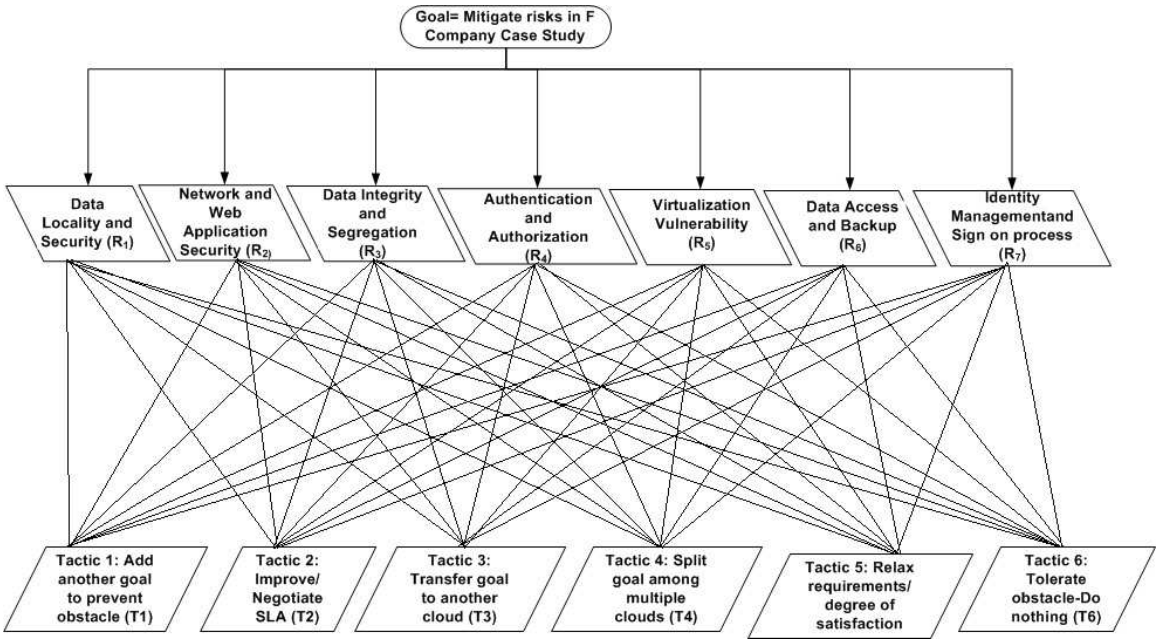

Figure 7.2: Hierarchy for Company F

and R_2, for example) insert their determined relative priority in the position (R_1,R_2) where the row of R_1 meets the column of R_2. In position (R_2, R_1) insert the reciprocal value. Continue to perform pairwise comparisons of R_1-R_3, R_1-R_4, and so on. Thus for prioritizing the risks 21 pairwise comparison are required. We will first populate the upper triangle with the weights

$$\begin{array}{c} \\ \mathrm{R_1} \\ \mathrm{R_2} \\ \mathrm{R_3} \\ \mathrm{R_4} \\ \mathrm{R_5} \\ \mathrm{R_6} \\ \mathrm{R_7} \end{array} \begin{array}{c} \begin{array}{ccccccc} \mathrm{R_1} & \mathrm{R_2} & \mathrm{R_3} & \mathrm{R_4} & \mathrm{R_5} & \mathrm{R_6} & \mathrm{R_7} \end{array} \\ \begin{pmatrix} 1 & 7 & 3 & 3 & 7 & 5 & 9 \\ & 1 & 0.33 & 0.20 & 4 & 4 & 9 \\ & & 1 & 0.33 & 5 & 5 & 9 \\ & & & 1 & 7 & 7 & 9 \\ & & & & 1 & 0.33 & 3 \\ & & & & & 1 & 3 \\ & & & & & & 1 \end{pmatrix} \end{array}$$

The lower triangle will be filled with the reciprocal values of the upper triangle. E.g. if 3 was inserted at position (R_1, R_3) then (R_3, R_1) will have the reciprocal value i.e 1/3. Pairwise

comparison of risks according to their criticality on the system gives the following matrix

$$\begin{array}{c} \\ R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} R_1 & R_2 & R_3 & R_4 & R_5 & R_6 & R_7 \end{array} \\ \begin{pmatrix} 1 & 7 & 3 & 3 & 7 & 5 & 9 \\ 0.14 & 1 & 0.33 & 0.20 & 4 & 4 & 9 \\ 0.33 & 3 & 1 & 0.33 & 5 & 5 & 9 \\ 0.33 & 5 & 3 & 1 & 7 & 7 & 9 \\ 0.14 & 0.25 & 0.20 & 0.14 & 1 & 0.33 & 3 \\ 0.20 & 0.25 & 0.20 & 0.14 & 3 & 1 & 3 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.33 & 0.33 & 1 \end{pmatrix} \end{array}$$

The priority obtained for the risks after the comparison is:

$$\begin{array}{c} \\ R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{array} \begin{array}{c} \begin{array}{cc} \textit{Priority} & \textit{Rank} \end{array} \\ \begin{pmatrix} 0.38 & 1 \\ 0.096 & 4 \\ 0.158 & 3 \\ 0.267 & 2 \\ 0.032 & 6 \\ 0.048 & 5 \\ 0.019 & 7 \end{pmatrix} \end{array}$$

Taking the sum of the components of the ratio vector and dividing it by the number of components we have an approximation to a number λ (called the principal eigenvalue). The closer the λ is to n (the number of activities in a matrix) the more consistent is the result. The deviation from consistency is represented by (λ-n)/ (n-1) which is the consistency index. The ratio of consistency index to random index is called the consistency ratio. The values of the random index for matrix of different orders are defined in [131]. The consistency ratio of 0.10

or less is considered acceptable. For risks the consistency ratio is acceptable.

$\lambda = 7.780$

$CI = 0.130$

$CR = 0.097$

7.1.5 Select an appropriate risk mitigation tactic

Since we equate risks with obstacles, we compare all the five obstacle resolution tactics described in Table 5.2 with respect to every risk for selecting the best resolution tactic.
We calculated the priority of obstacle resolution tactics for all seven risks using AHP. The priority of tactics for every risk is:

$$\begin{array}{c} \\ \mathrm{R}_1 \\ \mathrm{R}_2 \\ \mathrm{R}_3 \\ \mathrm{R}_4 \\ \mathrm{R}_5 \\ \mathrm{R}_6 \\ \mathrm{R}_7 \end{array} \begin{array}{c} \begin{array}{ccccc} T1 & T2 & T3 & T4 & T5 \end{array} \\ \begin{pmatrix} 0.112 & 0.497 & 0.277 & 0.032 & 0.082 \\ 0.51 & 0.073 & 0.264 & 0.039 & 0.113 \\ 0.524 & 0.125 & 0.247 & 0.036 & 0.069 \\ 0.519 & 0.137 & 0.249 & 0.028 & 0.067 \\ 0.49 & 0.15 & 0.264 & 0.03 & 0.066 \\ 0.566 & 0.124 & 0.228 & 0.028 & 0.053 \\ 0.543 & 0.25 & 0.113 & 0.028 & 0.066 \end{pmatrix} \end{array}$$

7.1.6 Stability and Sensitivity Analysis

The selection of risk mitigation tactics is a multicriterion decision problem. Assignment of weights to the risks and resolution tactics depends on human judgment. We will see whether the results are stable after slight changes in the assigned weights. We use equations 6.1 and 6.2 for calculating the stability index.
In order to calculate the stability of our solution, we put the priority vectors in a matrix for

every risk:

$$\begin{array}{c} \\ R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{array} \begin{array}{c} \begin{array}{cccccc} Weight & T1 & T2 & T3 & T4 & T5 \end{array} \\ \begin{pmatrix} 0.38 & 0.112 & 0.497 & 0.277 & 0.032 & 0.082 \\ 0.096 & 0.51 & 0.073 & 0.264 & 0.039 & 0.113 \\ 0.158 & 0.524 & 0.125 & 0.247 & 0.036 & 0.069 \\ 0.267 & 0.519 & 0.137 & 0.249 & 0.028 & 0.067 \\ 0.032 & 0.49 & 0.15 & 0.264 & 0.03 & 0.066 \\ 0.048 & 0.566 & 0.124 & 0.228 & 0.028 & 0.053 \\ 0.019 & 0.543 & 0.25 & 0.113 & 0.028 & 0.066 \end{pmatrix} \end{array}$$

The α power mean aggregated values for the different tactics for values of α of 2, 3 and 5 are shown in Table 7.1.

Table 7.1: α power mean aggregated values for different tactics for Company F

α	T1	T2	T3	T4	T5
2	0.4167	0.3232	0.2588	0.0321	0.0780
3	0.4462	0.3638	0.2597	0.0323	0.0794
5	0.4750	0.4098	0.2611	0.0327	0.0827

We consider two situations: potential changes of up to ϵ=100% of the alternative values and potential changes of up to ϵ=10% of the alternative values. We calculate the stability values for the order for each pair of consecutive alternatives $\delta_j(\epsilon)$ and then the stability index S(ϵ). Table 7.2 shows the calculated stability index.

The stability index is highest for α 2 and 3 for variations of upto 10%. The stability index is highest for variations of up to 100% when α is 5. This suggests that overall solution is best when α is 3.

7.1.7 Risk Mitigation using Prioritized Tactics

R_1**:** The company is concerned about the locality of data. The Salesforce has a number of options to choose from for the location of data. Hence the prioritized tactic i.e. negotiation of SLA can aptly mitigate the risk.

Table 7.2: Stability index for risk mitigation tactics for Company F

α	Stability Index S(0.1)	Stability Index S(1)	δ (1) Priority(tactic)			
			1(T1)	2(T2)	3(T3)	4(T5)
2	1	0.23	0.12	0.11	0.53	0.41
3	1	0.24	0.1	0.16	0.53	0.42
5	0.92	0.26	0.07	0.29	0.51	0.43

R_2**:** We add a goal to to mitigate risk R_2. Salesforce provides a tool called *Burp Suite* which can be used to evaluate the security of web applications. Additional security can be achieved by the use of firewalls, anti-malware etc.

R_3**:** The integrity and quality of data is of utmost importance when adopting the CRM services from a SaaS provider. The user can define the mandatory and optional fields in the table. The user can also use the Task Validation Rules by Salesforce. This will allow users to be more sophisticated with their data entry, such as ensuring that the start date is before the end date, number of units sold does not exceed the total inventory etc.

R_4**:** The Salesforce can provide a delegated authentication process to the customers. This allows users to integrate authentication with their LDAP (Lightweight Directory Access Protocol) server. This will result in greater security as Company F can differentiate from all other companies that use Salesforce in order to reduce phishing attacks.

R_5**:** It is recommended by Salesforce to test their online services for all the vulnerabilities to minimize the risk to the user data. Salesforce openly asks their customers to report the vulnerability and they would ensure that the security team provides the estimated time to fix that vulnerability and notifies the customer when it is fixed.

R_6**:** Company F can generate the backup files of its data on a weekly or monthly basis. The exported files will store data in a set of comma-separated CSV files.

R_7**:** Through identity and access tools provided by the Force.com platform a developer is allowed to fully automate user provision. User profiles can be created using a template. Field accessibility can be limited to certain profiles by the administrator, if required. Connection through external applications can be established through a security token. In short, users can be created, disabled, signed in and managed in many ways using Force.com platform.

7.2 Case Study 2

7.2.1 Myki System

We looked at the real world case study of “myki” smartcard system [45]. Myki is a contactless smartcard system created to automate and harmonize ticketing on all modes of public transport (trains, buses and trams) in Victoria which is the most populated state in Australia. 547 million passengers were reported to have boarded on all modes of transportation in 2010 and Melbourne has the largest tram system in the world [45]. According to the authors in [45] since Victoria is one of the most livable places in the world, future improvements in the system will be more and more important to all the stakeholders. Due to the possible future improvements, "myki" may be a possible candidate for cloud adoption. We use this case study as we are provided with some basic requirements and can refine the requirements into more concrete goals. We can also identify if there are any conflicting goals which may lead to risks. The activities associated with the Myki system from an end-user perspective are smartcard top-up, swiping card to enter vehicle, swiping card to leave vehicle (touch-off), and the automatic deduction and computation of fares. The system handles 800,000 requests every day. About 75% of the requests arrive during the peak hours and are equally divided between the evening and morning rush hours. The rush hours for morning are between 0700-0930 and evening hours are between 1700-1900. By applying Little’s law [106] the authors in [45] have calculated that the number of requests that should be handled by the cloud simultaneously is 2500. For the Olympics and nationwide adoption of the cloud we can calculate the number of requests

to be handled using the similar calculations. The authors in [45] think that the myki system could be a possible candidate for migration to cloud due to the high number of people using the system.The authors make two assumptions that

1. Melbourne earns the right to hold the Summer Olympics
2. The Federal Government of Australia decides to adopt “myki" for all public transport in Australia.

It is assumed by the authors in [45] that from the current workload the money generated is $100,000,000. The sole revenue source is the fare collection. It is assumed that the cost on cloud should not exceed more than 5% of total revenue in peak, current and Olympics scenarios and 50% in nationwide workload.

These futuristic assumptions will probably lead the decision makers to consider whether proposed designs will scale according to the workload. Goals related to profitability and performance will also become more important as design considerations. Profitability goals can be in conflict with performance goals. Some goals may be essential for achieving good performance but may affect the scalability goals and therefore may be considered obstacles. For example, users demand higher performance levels while keeping expenditures low, the cost to acquire a cloud service may increase. In this chapter we focus on scalability and performance goals and show how our approach is helpful in mitigating the risks by resolving the obstacles using obstacle resolution tactics. Figure 7.3 shows the goals essential for achieving performance and scalability of the myki system. However, we can see that some goals can act as an obstacle in the system.

7.2.2 Goal Tree of the myki System

Figure 7.3 shows the goal tree of the myki system. The scalability and performance goals are further refined into subgoals. The goals which point towards the parent goal are contributing to the satisfaction of the parent goals. It is evident from the goal tree that the goal Scale Horizontally [Data center] is essential for satisfying one goal but it acts as an obstacle for three other goals in the system. It is evident from the goal tree of the myki system some

goals may adversely effect other goals in the system. Scaling Horizontally [Data Center] may result in adversely effecting the goals High Throughput [Requests], Fast Response Time [Top up] and Fast Response Time [Touch Off]. High Utilization [Data Center Servers] may serve as an obstacle to Fast Response Time [Data Center]. Scale Vertically [Data Center] may be an obstacle for Minimize Cost [Cloud Service]. Hypothetical Traffic Burst (Data Center) may be an obstacle for goal Fixed Budget [Cloud Service].

Table 5.2 shows some obstacle resolution tactics proposed earlier in [179]. Obstacles if not resolved may turn into risks. To resolve the obstacles we prioritize the obstacle resolution tactics using AHP.

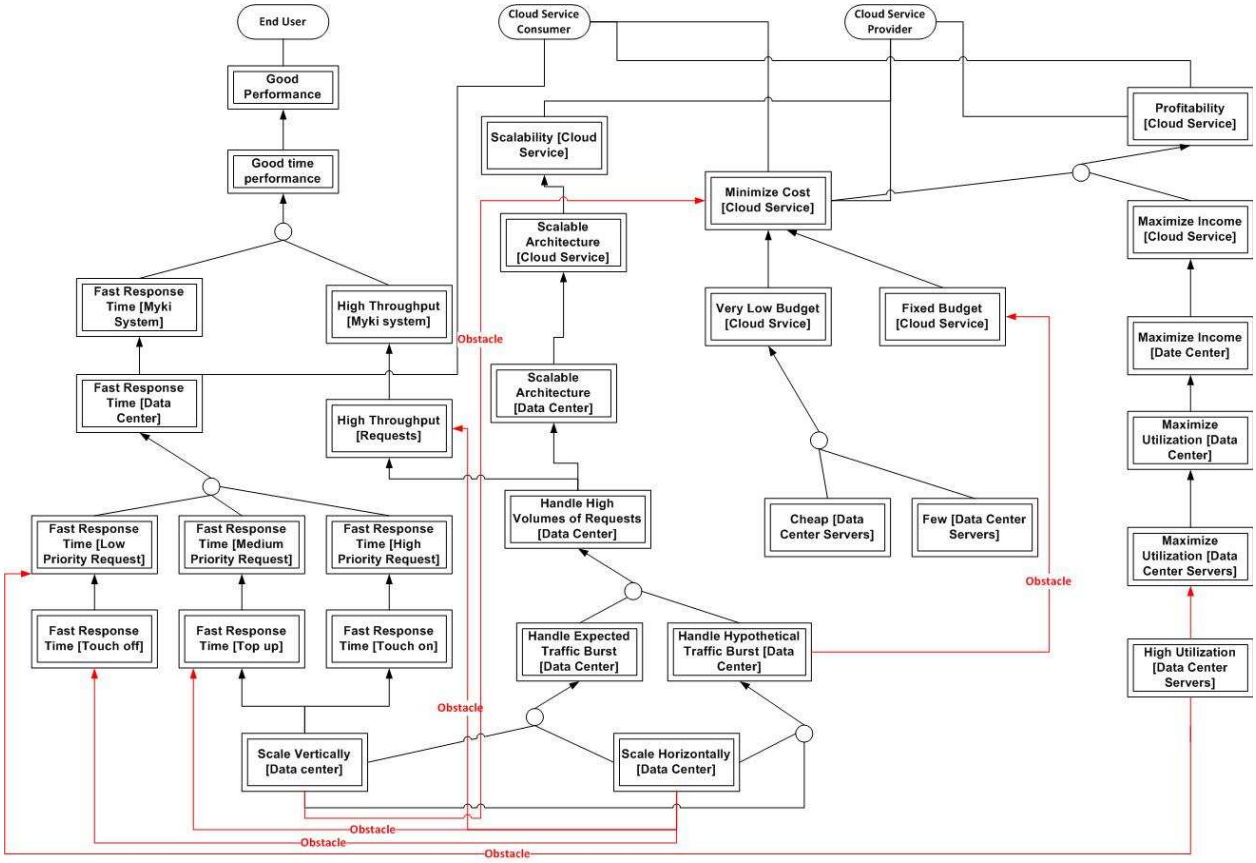

Figure 7.3: Goal tree of myki

7.2.3 AHP hierarchy for myki system obstacles

Figure 7.4 shows the AHP hierarchy of myki obstacles and the obstacle resolution tactics. For the purpose of simplicity we write O_{m1} for obstacle Scale Horizontally [Data Center], O_{m2} for obstacle Scale Vertically [Data Center], O_{m3} for obstacle High Utilization [Data Center], O_{m4} for obstacle Hypothetical Traffic Burst [Data Center]. The goal in our AHP hierarchy is to resolve the obstacles in the myki system. The second level of the hierarchy has the obstacles. The third level of hierarchy has all the alternatives for resolving the obstacles in the myki system.

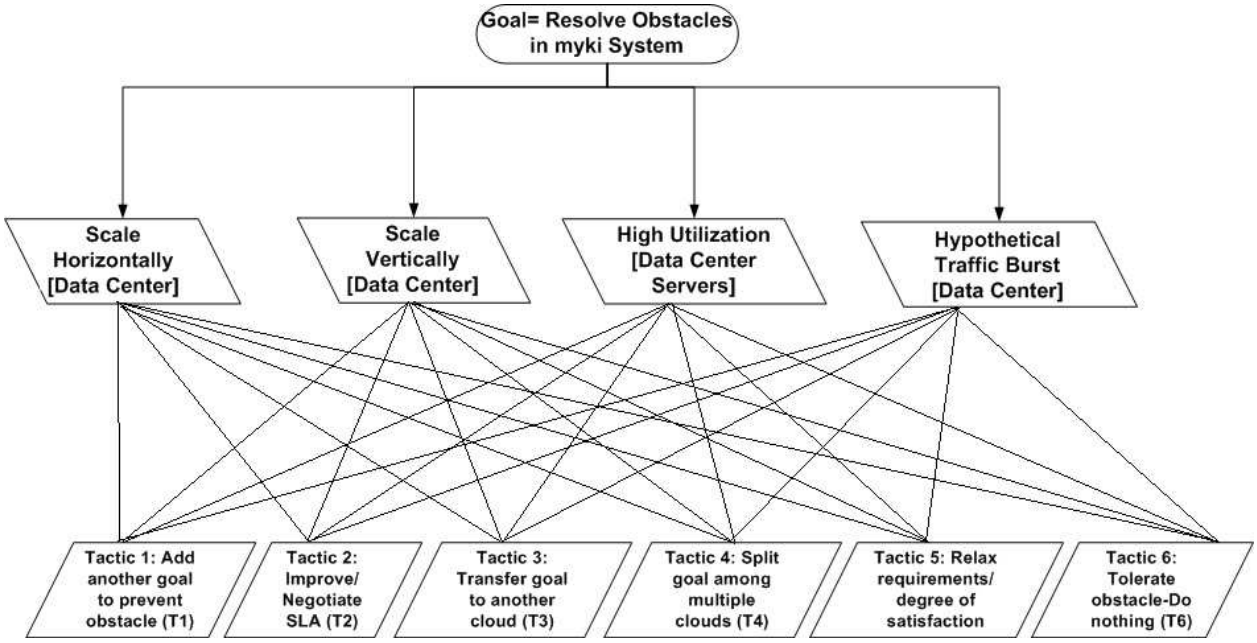

Figure 7.4: AHP Hierarchy of myki Obstacles

7.2.4 Prioritize Obstacles Using AHP

Prioritize the obstacles by pairwise comparisons using Table 6.1. Since there are four obstacles, the matrix is of order 4x4. Total number of comparions is 6. Pairwise comparison of obstacles according to their criticallity in the system gives the following matrix:

$$\begin{array}{c} \\ O_{m1} \\ O_{m2} \\ O_{m3} \\ O_{m4} \end{array} \begin{array}{c} \begin{array}{cccc} O_{m1} & O_{m2} & \mathrm{Om}_3 & O_{m4} \end{array} \\ \begin{pmatrix} 1 & 3 & 3 & 5 \\ 0.33 & 1 & 3 & 5 \\ 0.33 & 0.33 & 1 & 3 \\ 0.20 & 0.20 & 0.33 & 1 \end{pmatrix} \end{array}$$

The calculated priority of the obstacles is:

$$\begin{array}{c} \\ O_{m1} \\ O_{m2} \\ O_{m3} \\ O_{m4} \end{array} \begin{array}{c} \begin{array}{cc} \mathit{Priority} & \mathit{Rank} \end{array} \\ \begin{pmatrix} 0.505 & 1 \\ 0.288 & 2 \\ 0.143 & 3 \\ 0.064 & 4 \end{pmatrix} \end{array}$$

$$\lambda = 4.198$$

CI= 0.066

CR= 0.073

Consistency ratio is in the acceptable range.

7.2.5 Select an appropriate obstacle resolution tactic

We prioritize the obstacle resolution tactics for all four identified obstacles. We only prioritize five obstacle resolution tactics for every obstacle. If the cloud consumer does not think any of them is suitable or investing in obstacle resolution is a costly business, he may opt to tolerate the obstacle. The priority of tactics with respect to every obstacles is

$$\begin{array}{c} \\ O_{m1} \\ O_{m2} \\ O_{m3} \\ O_{m4} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathit{Weight} & T1 & T2 & T3 & T4 & T5 \end{array} \\ \begin{pmatrix} 0.505 & 0.545 & 0.226 & 0.127 & 0.068 & 0.035 \\ 0.288 & 0.06 & 0.54 & 0.265 & 0.097 & 0.038 \\ 0.143 & 0.041 & 0.551 & 0.26 & 0.064 & 0.084 \\ 0.064 & 0.032 & 0.478 & 0.314 & 0.114 & 0.062 \end{pmatrix} \end{array}$$

7.2.6 Stability and Sensitivity Analysis

The α power mean aggregated values for the different tactics for values of α of 2, 3 and 5 are shown in Table 7.3.

Table 7.3: α power mean aggregated values for different tactics for myki

α	T1	T2	T3	T4	T5
2	0.389024863	0.4096487508	0.2105870105	0.0803889545	0.0478489394
3	0.4341350965	0.4346076561	0.2216457749	0.0821274734	0.0516095854
5	0.4753961343	0.4677107532	0.2376679263	0.085590012	0.0588522756

We calculate the stability values for the order for each pair of consecutive alternatives $\delta_j(\epsilon)$ and then the stability index S(ϵ). Table 7.4 shows the calculated stability index.
The stability index is highest for α 2 for variations of upto 10% and 100%. The stability index is highest for α 2 and 3 for variations of upto 10%. The stability index is zero for variations of upto 100% and 10% when α is 5. This suggests that overall solution is best when α is 2.

7.2.7 Obstacle Resolution Using Prioritized Tactics

O_{m1}: One central database may be used where virtual machines can communicate with it via a load balancer. Use a second database for failover.
O_{m2}: Negotiate the SLAs to scale vertically. As the traffic during peak hours is different from the off peak hours the SLA should be negotiated so that system can scale up and scale down accordingly.
O_{m3}: High utilization of data centre servers improves the profitability however, it hurts fast response time. Negotiate SLA (by looking at the benchmarks) to reduce the chance of hurting goals due to high utilization. Server utilization can be given a boundary value like utilization should be between 20 to 40%.
O_{m4}: Negotiate with the cloud service provider by providing them the details of hypothetical traffic bursts (e.g. how much traffic is expected during Olympics). Cloud should be able to scale up according to the traffic. If the cloud is not engaging into negotiation then probably myki can shift to another cloud when the traffic is large.

Table 7.4: Stability index for obstacle resolution tactics for myki

α	Stability Index S(0.1)	Stability Index S(1)	δ (1) Priority(tactic)			
			2(T1)	1(T2)	3(T3)	4(T4)
2	0.707107	0.16855	0.025	0.29	0.44	0.253
3	0.173205	0.049774	0.0002	0.31	0.45	0.22
5	0	0	0.0	0.33	0.47	0.18

7.3 Qualitative Evaluation and Discussion

This chapter has presented two case studies to assess the effectiveness of our method for mitigating the risks for cloud adoption. The objective of these case studies is to exemplify the use of our method in the process of cloud adoption. We have evaluated our framework for mitigating the risks associated with cloud adoption using GORE and AHP. We prioritized the risks and risk mitigation tactics using AHP. We did the stability analysis of the AHP choice of the mitigation tactics.

We believe that Company F case study has proved to be a valid evaluation attempt. We modeled the obstacles/risks in goal-oriented form which Company F considered to be most important. Our method enabled us to see whether salesforce.com can actually be adopted while mitigating the identified risks. This has ensured that decisions regarding SaaS adoption are made on objective and clear arguments. The calculated results of the AHP choices of the resolution tactics are stable. This method has informed users how they can mitigate the risks while adopting salesforce.com SaaS. We did the matching between the prioritized tactics and the availability of methods in the Salesforce domain which could help in mitigating the risks. We found that Salesforce was able to address the concerns of Company F and therefore we deem Salesforce suitable for Company F.

Obstacles in myki case study were largely related to scalability . The obstacles in the myki case study were identified from the goal tree and obstacle resolution tactics were proposed for the identified obstacles after prioritizing the obstacles and obstacle resolution tactics using AHP. GORE has been used for modeling requirements in the myki case study. While some subgoals were contributing to the achievement of parent goals, they were contributing negatively to the achievement of other goals. Such inconsistencies between goals were identified and

the obstacles were resolved through obstacle resolution tactics. Small perturbations in the assignment of weights did not effect the priorities of the resolution tactics for mitigating the risks. Table 7.4 and table 7.2 show the stability indices for myki and Company F's case studies. Table 7.5, table 7.6, table 7.7 and table 7.8 show the evaluation for each phase of the method for its completeness, subjectivity and scalability. We now present the answers to the evaluation criteria defined at the beginning of this chapter.

Why is goal-oriented a suitable requirements engineering method for specifying user requirements for cloud adoption?

Our interest in goal-oriented requirements engineering is based on the fact that goal-oriented requirements engineering starts from the high level goals which are refined to more concrete goals. This allows traceability links from high-level to low level goals [153]. The goal graph allows us to see how satisfaction of one goal may hurt the satisfaction of another goal, and therefore goal-oriented modeling helps in identifying the conflicting requirements. As we have discussed in chapter 2, requirements keep changing in cloud context as compared to classical system development. Goals are considered to be more stable [153] and therefore it is reasonable to use goals for requirements specifications.

Its imperative that requirements can continue to change in cloud context. As a result GORE can provide better support for evolution where we can focus the analysis on part of the goal tree which has been directly or indirectly effected by the modification in SLAs or user requirements. Consequently, the analysis could be repeated to discuss the service ability to meet the changes in the requirements. Though this thesis hasn't focused on scenarios related to change and evolution we argue that the method could be adapted easily to support such evolution. GORE is a sound methodology which has been widely adopted in different application domains. The method has well defined process for refinement, elaboration, specification and operationalization of goals. The method is complemented with obstacle analysis and resolution which render themselves to useful technique for our context. The

implication of the choice of this method is that it can ease comprehension (due to its wide use the terminologies are defined and we have a language to define the goals and obstacles). It's flexible enough to be customized to various solution domains such as clouds. It explicitly supports the traceability of the decisions made and support for change and evolution. The tree structure also supports scalability.
Case Study 2 has used goal-modeling for requirements specification. Through the goal graph we were able to identify interactions between different goals. Where we have only high level goals provided we can refine them into more concrete goals in the light of cloud provisions. This strategy of refining goals is helpful when we have very few goals to begin with and evaluation of the cloud is to be done together with goal refinement. For the above mentioned reasons we can say that goal-oriented RE is a suitable method for acquiring and specifying user requirements.

How are the obstacle identification and obstacle resolution techniques useful for mitigating the risks associated with cloud adoption?

In Case Study 1 the risks were already defined by the company. We modeled the risks in goal-oriented form. We proposed ways for mitigating those risks using obstacle resolution tactics. In Case Study 2 the obstacles were identified after modeling requirements in a goal-oriented form. We identified how a subgoal can contribute in the achievement of one goal and be an obstacle for another. After identifying the obstacles we proposed obstacle resolution tactics for every identified obstacle. Since obstacles can be indicative of risks [179] therefore resolving obstacles through the proposed resolution tactics will help in mitigating the risks associated with cloud adoption. The exercise of applying the resolution tactics to resolve obstacles for mitigating risks has been demonstrated to be valid.

How can the method be useful for identifying the conflicts and tradeoffs in user requirements?

Conflicts need to be identified and resolved at the RE time in order to get consistent requirements [153]. In Case Study 2 it is evident that while some subgoals contribute to the achievement of one goal they may hurt another goal. For example, scaling horizontally may help in achieving the larger goal of scalability however it may hurt performance (fast response time). The method therefore helps in identifying the conflicting requirements and tradeoffs. Hence, a goal tree is helpful in not just identifying but also managing the conflicts. The conflicts can be resolved through negotiation of the requirements or through alternative agent selection [153]. We have tried to manage the conflicts and tradeoffs using the obstacle resolution tactics.

How can we determine the criticality of the risks through prioritization? Why is the choice of AHP for prioritization justified?

It is important to prioritize the risks according to their impact. Not all risks will have a significant adverse impact if they occur. Some risks can be ignored whereas it is important to concentrate on the more critical risks [79]. We use AHP to determine the priority of risks through pairwise comparisons. After setting the priorities the users can decide which risks are critical and should be mitigated before moving to the cloud. A risk may be mitigated through several ways. We have therefore used AHP to prioritize the risk mitigation/obstacle resolution tactics so that users make informed decisions about the way to mitigate the risks. Risk mitigation tactics have been proposed for both case studies after prioritizing the risks and risk mitigation tactics for each risk.

We have used AHP and utility theory for prioritization. In chapter 5 we used utility theory for goal and obstacle prioritization. However, we later felt that since weight assignment is subjective therefore we need to have a scale where a given weight has a definite meaning. Utility and AHP are both helpful in complex decision making however there are many differences. According to Saaty [132] there have been some major problems with utility theory identified by the experts. [132] explains that the main difference between AHP and utility theory is that AHP is based on pairwise comparisons. Utility theory on the other hand rates alternatives

one at a time [132]. We therefore prefer AHP for prioritization because it provides users a scale from which they can assign values to the alternatives. Since every number assigned to an alternative has a meaning, it becomes easier for different stakeholders to use AHP. Since the assignment of weights are based on human judgment, AHP also has a method to check the consistency of the results. Utility theory however does not check whether the results are consistent or not. Inconsistencies are therefore dealt with in a formal manner in AHP [75]. Our method is also supported by the stability analysis of the results obtained from AHP prioritization.

Does the priority of obstacles and obstacle resolution tactics remain the same after small perturbations in values of the alternatives?

We calculate the priority of obstacles and obstacle resolution tactics using AHP. Priorities are assigned through pairwise comparison using ratio scale presented in table 6.1. Since the priority of the obstacles and obstacle resolution tactics is determined through human judgment we need to do the stability analysis of the results. By stability we mean that the priorities will not change if there are small perturbations in the assignment of weights to the obstacles and obstacle resolution tactics. We have performed the stability analysis for both the cases and have found that results are reasonably stable.

Scalability

There are several dimensions for discussing the scalability of the method. Among these dimensions are:

- *Stakeholders Inputs:* As the number of stakeholders' input increase, the analysis may become more complex. This is likely to result in more perspectives, more conflict to resolve, more negotiation to take place. The situation can become serious in situations where stakeholders are not identified based on relevance and experience. It is important to identify and prioritize the stakeholders who should be involved in the whole process

of identifying a suitable cloud. Not all stakeholders are equally influential and to identify the influential stakeholders there are tools available like StakeNet [105]. This will limit the number of stakeholders involved in applying the method.

- *Goals/Requirements Elicitation:* Scalability also becomes an issue if the goal tree becomes too big due to the large number of subgoals, we can prune the tree to make the prioritization process easy. By pruning the tree we mean that all requirements arising from one particular node are separately prioritized. E.g security subgoals are separately prioritized from the performance subgoals. Requirements can also be grouped according to their priority as proposed by [86].
- *AHP/Pairwise Comparisons:* AHP requires pairwise comparison for prioritization. Redundancy makes this process very robust. However, scalability becomes an issue in AHP's application to goal models [102]. According to [21] the maximum number of pairwise comparisons a human can make with consistency is about 100. [171] use the clustering technique to reduce the number of comparisons. The clustering process divides objects into different subsets so that objects in each group have similar properties. [86] has proposed grouping the requirements to reduce the number of comparisons. We believe that GORE and handling of obstacles through GORE provides an elegant grouping benefiting from the goal refinements hierarchy, promoting potential scalability.
- *Number of Tactics:* The number of tactics for resolving the obstacles may become large due to the stakeholders identifying new obstacle resolution tactics. It might be clear to the users that they may actually not use one particular tactic for resolving certain obstacle. If the stakeholders are sure of not using a certain tactic for resolving obstacle they can refrain from using that tactic for prioritization.

Subjectivity

Most of the selection decisions are based on subjective judgments.[7]. The decision to shortlist the cloud providers may not just be based on how well the cloud is able to meet the technical

requirements. Other factors, subjective in nature, may also influence the cloud selection decision. For example a technically better cloud may be avoided due to monetary constraints. There are some goals which are subjective by nature and cannot be measured by numerical metrics [9]. Goals are elicited by different methods as defined in chapter 3. Due to the subjective nature of the goals there may not be a clear-cut definition of goal satisfaction. It is therefore good to define a goal satisfaction level [177].

A distinctive feature of AHP is the amount of subjectivity used. AHP elicits the opinion using pair-wise comparison. The subjective judgment, selection and the priority given by the decision-makers have significant influence on the AHP method [43]. We are using the feature analysis method of DESMET [91] which is based on judging methods against some kind of " evaluation criteria" which are subjectively identified. Such evaluations are therefore prone to bias [90]. The assignment of weights due to subjective criteria are approximate. It is therefore important to do the stability analysis of the results obtained from AHP. Results are considered to be stable if the order of the solution does not change with slight perturbations in the values of the alternatives. Hence we have performed the stability analysis in both the case studies and have obtained stable results.

Completeness

Achieving a complete set of requirements is a major concern in RE [153]. Goals set criteria for requirement completeness. The requirement specification for a particular set of goals is complete if all the goals can be achieved from the specification [176]. Goals in Case Study 2 were refined until they can be assigned to the agents and obstacles were identified to ensure the completeness of the requirements. Since obstacles can hinder the goal satisfaction, it is also important to achieve completeness for obstacles for all the goals. In Case Study 1 the risks were already identified however in Case Study 2 we have identified obstacles which may hinder the fulfillment of some goals. Chapter 3 has defined conditions for requirements and obstacle completeness. We believe that the case studies used for evaluation have shown how negating an obstacle will result in the satisfaction of a goal hence ensuring requirement

completeness.

Stakeholder Identification and AHP Inputs

The values to the obstacles and resolution tactics can be provided by the users through a voting procedure where stakeholders can assign weights based on values in table 6.1. Stakeholders who can be involved in the voting procedure can be identified from already existing tools like StakeNet [105]. Since numbers assigned for determining the priority are coming from different stakeholders, the aggregate of the priorities assigned by different stakeholders may give us the overall ranking [4]. A platform named StakeCloud which acts as a marketplace has been proposed [145]. Users can input their requirements and these will be mapped to the service offerings of the cloud providers.

Early Payback

According to [47], methods and tools should show significant benefits as soon as people start using them. Early payback is the main objective of this method. As we discussed in chapter 1, in terms of our motivation for this work, our aim is to screen the cloud according to user requirements to mitigate the risks during the early stages. Getting locked in with a cloud which is unable to meet user requirements may result in significant financial losses. Our method will help them in making informed decisions about cloud selection. From the Case Study 1 it is evident that the risk mitigation tactics for all the potential risks were proposed specific to the Salesforce.com SaaS. Obstacles were identified from the goal tree of Case Study 2. The obstacle resolution tactics were proposed for the identified obstacles. This early identification of risks, obstacles and mismatches between user requirements and cloud provisions can save users from investing in the wrong cloud.

Table 7.5: Identification of the Stakeholders

Phase	Subjectivity/Bias	Completeness	Scalability	Systematic Vs Adhoc
Identifying the stakeholders	Identifying the stakeholders is critical. Stakeholder identification is subjective based on their relative power and influence [65]. Not all stakeholders are equally influential we therefore need to identify the stakeholders for cloud RE. [69] has proposed that stakeholders should be prioritized by assessing the risk that might be incurred from neglecting them. There may be tensioning of relationships between different stakeholders due to their different priorities [183]. It is therefore important to resolve the tensions during the RE phase. From case study 2 it's evident that there may be tensions between different goals of different stakeholders. E.g. fixed budget may be an obstacle for hypothetical traffic burst. We have applied our method to resolve these tensions using obstacle analysis, obstacle prioritization and resolution.	While stakeholders may be identified from existing tools like StakeNet [105], completeness cannot be ensured. Due to the dynamic evolution of the services we cannot ensure the completeness of identified stakeholders. For example, a cloud may update their SLA and include a new stakeholder by subcontracting their data center.	When the number of stakeholders increase the adoption process of cloud becomes more complex. Since not all stakeholders are equally important therefore we can use the existing tools like StakeNet [105] for identifying the most influential stakeholders for cloud adoption. This will minimize the efforts on resolving the conflicts resolution, tradeoffs analysis, negotiation etc.	Our method is based on GORE. Goals are refined and a goal tree is drawn until the goals cannot be further refined. This is a systematic way of identifying the stakeholders who may be involved in the process of cloud adoption. Goals can be traced to their origin by following the goal graph. Our proposed method therefore provides a systematic way of identifying the stakeholders instead of randomly selecting the stakeholders for cloud adoption. It can be seen in Case Study 2 the goals are tracing back to their origin (i.e the stakeholders)

Table 7.6: Requirements Elicitation

Phase	Subjectivity/Bias	Completeness	Scalability	Systematic Vs Adhoc
Requirements Elicitation	Since goals can be influenced by cloud SLAs; therefore, the elicitation process can be biased. Requirements bias can also be influenced by the choice of stakeholders, expertise, personal bias to the reputation and perception of the provider [177] The influential [104] stakeholders may effect the requirement elicitation process.	Requirements elicitation for cloud cannot be complete due to the frequent evolution of cloud services. However, the use of GORE and obstacle analysis ensure the completeness of requirement specification. Completeness is subject to the probable obstacles which obstruct the goal and our consequent handling of these obstacles. That is, completeness is highly dependent on our knowledge and of the domain and our expertise [163], their likelihood of occurrence and their consequence. Nevertheless, GORE provide built-in support for completeness through continuous refinements, elaboration, monitoring for goal satisfaction [153]	If there are large number of goals and the goal graph becomes extremely big it is better to prune the tree to make prioritization of requirements easy. Requirements arising from one node of the tree are separately prioritized .	We have equipped users with a framework which helps them in gathering requirements in a systematic way. Previously the cloud adoption was based on adhoc inputs. The requirements can be documented for legal bindings and subsequent followup, maintenance and evolution. The whole exercise will also help users to make informed decisions. This will bring more transparency and will promote accountability in clod adoption.

Table 7.7: Obstacle Identification and Resolution

Phase	**Subjectivity/Bias**	**Completeness**	**Scalability**	**Systematic Vs Adhoc**
Risk/Obstacle identification and mitigation	Risk identification can be a subjective activity. Considering cloud as a black-box, users may also rely on their intuition for risk identification. Since obstacles can result into risks therefore it is important to identify and resolve the obstacles. It is preferred to identify what is *wanted* and to identify what might obstruct it [163]. The obstacles may be identified by considering different scenarios in which a goal may be obstructed and on past experiences [163]. They may also be identified from the SLA clauses unable to satisfy the defined goals [179].	Completeness of obstacle identification is highly dependent on the knowledge of the domain [163]. Risk/obstacle identification may not be complete due to incomplete information about the domain and insufficient information in the cloud SLAs. For example when the data is outsourced to a third party without making any such specification in the SLAs.	As the goal tree for the requirements grows so do the risks. The risks can also be clustered according to their type as proposed in [171] where authors have clustered different test cases together. For example, security related risks may be separately prioritized from the performance related risks. Not all risks may be critical and therefore we will have to prioritize them to determine which ones should be mitigated [179].	We have proposed a systematic method to identify the risks during the early adoption phase. We believe that risks should be dealt with in the requirements engineering phase so that they are not passed on to the actual system once the cloud services have been adopted. From case study 2 we have observed that if we had not modeled the requirements using GORE we would not have been able to identify the conflicting requirements. GORE has proved to be valid for obstacle and risk identification. Case study 1 has successfully used the obstacle resolution tactics for mitigating the risks.

7.4 Conclusions

We evaluated our proposed method by using two different case studies. While in case of the myki system we evaluated the PaaS services of the cloud, for Company F we evaluated the SaaS services. In the case study of Company F, the risks were already identified by the company's task force. Both the case studies have addressed different kind of risks. Company F's risks were related to security and identity management whereas myki's risks were related to scalability. By using the goal-oriented approach, we modeled the identified risks as obstacles and resolved them using the proposed obstacle resolution tactics. The risks were prioritized according to their criticality and were then mitigated. While a risk may be mitigated through different tactics, it is good to prioritize the risk mitigation tactics to select the best one [178]. Different phases of the proposed method were evaluated against the criteria of completeness, subjectivity and scalability. Table 7.5, table 7.6, table 7.7 and table 7.8 present the results of the evaluation.

From Case Study 2 we have learned that when we have only high level goals, we can refine them into more concrete goals in the light of cloud provisions. The goal graph also helped in identifying different kind of interaction among the goals. From Case Study 2 we have also learned that while some subgoals may be contributing towards the achievement of a parent goal, they might become a hurdle in achievement of another goal in the system. Goal-oriented modeling of requirements therefore helped in identifying the risks, conflicts and the tradeoffs. Not all risks may be critical, hence we prioritize the risks according to their impact. We have used AHP for prioritizing the risks and risk mitigation tactics. AHP makes pair-wise comparisons for prioritization which is an extensive process and hence more reliable. Scalability is an issue in case of AHP [21]. According to Karlsson et al. [86] the requirements can be grouped together to increase the number of comparisons. GORE provides an elegant grouping benefiting from goal-refinement hierarchy which helps in scalability [178]. The assignment of priorities is based on human judgment and is prone to errors. AHP allows us to check the consistency of the results [75]. We have been able to prioritize and mitigate the risks in both the case studies using AHP. We also did the stability analysis of the results. By stability analysis

we mean that the priorities remain unchanged if there are small changes to the assignment of weights. We have seen that the obtained results for both the case studies were sufficiently stable.

From table 7.5 table 7.6 table 7.7 and table 7.8 it is evident that the proposed method is scalable. Scalability has many dimensions which includes the increase in the number of stakeholders, their requirements, pairwise comparisons for prioritization and in the number of obstacle resolution tactics. When the stakeholders become too many the analysis may become very difficult. This will result in more tradeoffs, conflicts, negotiation etc. The stakeholders who should be involved in the process of cloud evaluation can be identified through prioritization. Already existing tools such as StakeNet [105] can be used for identifying the influential stakeholders.

Goal-oriented provides a systematic way of identifying the stakeholders as it provides the traceability links to the origin of goals. In Case Study 2 it can be seen that the goals are tracing back to their origin (i.e. the stakeholders). Our proposed framework also helps engineering the requirements for cloud adoption. There was no framework that could help users in screening and selecting the cloud according to their requirements while mitigating the risks. Our method not only helps users in eliciting the requirements but also identifying the risks during the early adoption phase. From Case Study 2 it is clear that we might have not observed some of the conflicting requirements had we not modeled them using GORE. GORE has therefore been useful in identifying the risks and obstacles. Case Study 1 and Case Study 2 have successfully used GORE for modeling, analysing and mitigating the risks using the obstacle resolution tactics. Our proposed method provides systematic way for acquiring user requirements and mitigating the risks associated with cloud adoption.

We have learned that keeping the goals flexible and negotiable is perhaps the best way to find the optimal cloud. When the goals are kept generic in the beginning, they don't restrict the user from evaluating a wide variety of cloud services on offer. The user can reject the cloud after or during evaluation when they see that the cloud is unable to satisfy the goals which cannot be compromised. Some of the goals may not be satisfied due to some obstacles.

We believe that any cloud service can be adopted as long as those obstacles can be resolved under the monetary constraints. Not all obstacles may be of critical nature and therefore it is important to prioritize them. We believe that priorities should remain same even if there are small changes in the assignment of priorities to the obstacles by the users. Therefore, once the obstacles and their resolution tactics have been prioritized it is good to see if the priorities remain unchanged when there are small changes in the assignment of weight. We have learned after evaluating these two different case studies that its good to mitigate the risks before passing them on to the final system. The identification and mitigation of risks is possible when we evaluate the cloud in the light of user requirements and see if there are any mismatches between user requirements and cloud provisions.

Table 7.8: Prioritization of Obstacles & Tactics

Phase	**Subjectivity/Bias**	**Completeness**	**Scalability**	**Systematic Vs Adhoc**
Prioritization of obstacles & tactics	AHP uses a scale which stakeholders can use to give weights to the obstacles and tactics based on their subjective expert input, experience and intuition [121]. Due to the subjective nature of AHP input there may be inconsistencies in assignment of weights. AHP provides a method for consistency checking [130]. Since assignment of weights is based on human judgment it is preferred to get stable results. Our methods supports the AHP choice of the resolution tactics using stability analysis [178].	AHP requires pairwise comparisons. Due to the exhaustive nature of AHP its unlikely that any risk is ignored. The pairwise comparison of risks therefore ensures that all risks are considered for prioritization and mitigation.	AHP requires pairwise comparison. Although the process becomes robust due to pairwise comparisons, scalability is an issue. Karlsson et al. [86] have recommended that requirements can be grouped together to reduce the number of comparisons. We believe that our GORE based methodology for cloud adoption and our handling of the obstacles in this process provide an elegant grouping benefiting from the goal refinement hierarchy which promotes scalability [178].	Not all risks may be critical, it is therefore important to prioritize the obstacles according to their impact and likelihood of occurrence. Our method provides a systematic way for identifying, prioritizing and mitigating the risks associated with cloud adoption. AHP helps in identifying the obstacles keeping in view multiple views. Due to excessive pairwise comparisons there is more room for identifying hidden risks. We resolve the obstacles by proposing obstacle resolution tactic for every obstacle after prioritizing the tactics for each one of them using AHP. Our method is supported by the stability analysis. We have observed that our results are reasonably stable.

CHAPTER 8

Conclusion and Future Work

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Winston Churchill

As our journey approaches an end, we now stop and look back at the path we took and the road ahead. This thesis has contributed to a novel framework that enables users to select cloud services according to their requirements while minimizing the risks.

We have looked at classical RE and its limitations for engineering requirements for cloud adoption. We have seen that requirements engineering for cloud is far more challenging as compared to classical requirements engineering, due to the ever changing cloud services and user requirements. Cloud requirements engineering needs to be more flexible in order to accommodate changes in user requirements and cloud services.

We have defined the steps involved in the requirements engineering phase for cloud adoption. Chapter 4 contributes to a novel lifecycle, which aims at providing systematic guidance for an organisation evaluating the choice and risks in moving to a cloud. We have used goal-oriented approach for eliciting and modeling the requirements of the user. We have presented systematic guidance for the identification and evaluation of cloud service providers. The evaluation process ends by selecting potentially "good enough" candidate cloud service provider. A key phase of the method is the matching phase where mismatches between the requirements and cloud service providers' features are identified. The analysis of the mismatches may inform the existence of risks. Our approach advocates risk mitigation in the early stages of cloud adoption. The approach also tries to manage the tradeoffs involved in moving to the cloud. As SLAs are largely too static at present, the SLAs do not address the individual needs of every user. This framework would help the cloud service provider and the user to negotiate their requirements and cloud features beyond static SLAs. It would help in making SLAs that are specifically designed for a particular organisation.

Chapter 5 contributes a lifecycle which is an extension of our previous work presented in chapter 4. The lifecycle can help in mitigating the risks which may arise due to incomplete requirements or insufficient cloud features. We have presented a systematic guidance for the identification of the obstacles and have proposed some concrete and abstract obstacle resolution tactics to minimize the risks associated with cloud adoption. We have also used utility theory to prioritized the goals for minimizing the risks.

Chapter 6 presents a systematic method for prioritizing the obstacles which may be encountered while adopting cloud services and the tactics for resolving them using AHP. The method is based on goal-oriented requirements engineering, obstacle handling and AHP to systematically identify, assess, prioritize and mitigate the risks. To see the applicability of our method we have exemplified the steps using a non-trivial case study. As the assignment of weights in AHP is based on human judgment, it is important to see that the results are stable even when small changes are made in the assignment of weights. By stability we mean that priorities remain the same. We have done stability analysis of the results and have found them to be fairly stable.

Chapter 7 has applied the proposed method on two non-trivial case studies. Each phase of the method has been evaluated against its applicability, scalability, completeness and subjectivity. Requirements from the case studies have been modelled using GORE which has resulted in identification of conflicting requirements and obstacles. We have prioritized the obstacles and tactics for resolving every obstacle. As a method it is highly human centered, we have discussed issues related to human bias, expertise, selection of stakeholders, interpretation of relevant artifacts like SLAs, skills of evaluators and sensitivity to inputs.

We have observed that modeling requirements using GORE conventions results in identifying the conflicts and tradeoffs among requirements. GORE also helps in identifying the stakeholders because of its traceability links to the origin of the goals. There may be some mismatches between user requirements and cloud features which may be indicative of risk. We have learned that it's better to mitigate the risk at the requirements engineering level than to get locked in with a cloud which cannot satisfy user requirements. Not all risk are of critical nature, therefore it is best to prioritize the risk according to their criticality. We have prioritized the risks using AHP. Since AHP does pairwise comparisons, therefore it is more reliable and less sensitive to judgmental errors. We believe that AHP and GORE provide an excellent way for grouping similar requirements together and then prioritizing them. Grouping the similar requirements together for prioritization helps in reducing the number of comparisons [86]. After evaluating our proposed framework we have come to the conclusion that the framework

is scalable. Although it cannot ensure completeness due to the black-box nature of the cloud, it makes the whole process of cloud adoption more transparent. Now the users are equipped with a framework which not only helps them in identifying and screening the cloud but also helps them in refining their own requirements. The whole process of cloud adoption was previously based on ad hoc inputs from different sources. Now, not only the possible risks can be identified but can be mitigated using the proposed tactics. We have also observed that negotiation of user requirements is one of the integral part of cloud adoption. We have therefore kept the goals generic in the beginning so that the potentially best clouds are not rejected in the beginning. Once the generic goals are acquired they can be refined in the light of cloud SLAs. We have also observed that SLAs are usually too static and can have some hidden risks despite all the assurances from the cloud service providers. After modeling the requirements we identified the risks which might have otherwise gone unnoticed. Our proposed method therefore provides a systematic way for acquiring and specifying the user requirements. It also helps in mitigating the risks associated with cloud adoption. If we don't engineer user requirements and make decisions based on ad hoc inputs, market reviews and recommendations, then the user will accept all the known and unknown risks associated with the cloud. Even if the cloud service provider does not engage in active negotiation, the user can evaluate the cloud and mitigate the risks through passive negotiation. So, even though the SLA will not be changed significantly by the cloud provider, the user can see if the SLA is able to satisfy user requirements and is not passing on any risks to the user.

Our proposed method has shown benefits by mitigating the risks in the cloud adoption process. However, the whole matching process between the cloud features and user requirements is heavily dependent on the availability of the information about the cloud. If there is a new cloud service and there is very little information available of that cloud then the matching process may not work fine because of less available information. Part of our proposed framework was used by Chung et al. [45] for evaluating the cloud services. We believe that there will be a significant number of users adopting our methodology for cloud evaluation and risk mitigation. Our research has been followed by different research groups from over the world

[1]. However, the process needs to be automated to make requirements elicitation, obstacles analysis, obstacles prioritization and resolution easier. In near future we plan to make a tool which would automate the whole process.

Future Work

Applying search-based techniques for cloud selection

We would like to complement our work with automated tool support using search-based techniques. The objective of search-based software engineering research is to move the human-centric software engineering search problems to machine-based search [76]. As some of the phases are human centric where scalability could be an issue, its possible to leverage on advances in search-based software engineering to tackle the complexity of the analysis and to optimize for multi-objective requirements. We would also like to see how information available through benchmarks could be used to improve the accuracy of matching and optimization for specific objectives.

Currently, our work has been heavily steered by the stakeholders' input however its possible to use inputs from benchmarks/published repositories and the effort would be how to mine them in order to find the suitable cloud.

Making decisions under uncertainty

The proposed method can allow us to do a what-if analysis. Stability analysis can help us test the impact of small changes in the input on the output. Although this approach can help us to manage uncertainty of inputs, its applicability tends to be limited and human centric. An effective and scalable what-if analysis should cater for wide variations of inputs and uncertainty revolving around them. It's possible to improve the analysis for making a decision [97] using the Monte-Carlo simulations. An automated approach for input generation could be possible. We can leverage on the work of Letier et al. [97] for making decisions under uncertainty using

[1]Our Paper " Cloud Adoption: A Goal-Oriented Requirements Engineering Approach" has so far been cited 46 times.

Monte-Carlo simulations.

Uncertainty makes early requirements complicated [97]. Although users are investing in the cloud under uncertainty due to its ever changing features and SLAs, there is a general lack of evaluating uncertainty, its impact on risk analysis and mitigation, and the value of reducing uncertainty before making a decision about cloud selection. We believe that the uncertainty about the cloud can have significant impact on stakeholders' goals. There is recent work [97] which takes into account uncertainty and calculates the impact of uncertainty through Monte-Carlo simulations. We intend to do the similar analysis regarding cloud selection using Monte-Carlo simulations to shortlist candidate clouds after taking into account cost, benefits and risks.

Viewpoint-based requirements engineering for scalability and conflict resolution

The current method is goal-oriented where obstacles are identified and prioritized for resolution. Different users may have a different perspectives about the same requirement. While a set of requirements may be of high importance for one stakeholder, for another it may be of less importance. A viewpoints-based requirements engineering approach accepts that system requirements cannot be obtained from a single perspective [139]. Due to the heterogeneity of cloud users we may have to organize requirements according to different stakeholders (we can classify the stakeholders as users, cloud providers,cloud consumer etc.) Viewpoints may help in organizing requirements coming from stakeholders having diverse backgrounds [139]. Its possible to focus the prioritization on the views and the goals which arise from these views [139]. We can build on viewpoints-based requirements engineering to benefit this proposal. Though viewpoints are effective in accommodating diverse inputs of stakeholders, their preferences and perspectives; it is imperative that inconsistencies, conflicts [93] and more negotiation could be the norm, however the extension is likely to result in more accurate analysis and stakeholder satisfaction.

Its possible to increase the scalability of the proposed technique using viewpoints. Viewpoints can give us diverse and heterogeneous view. We need to improve the accuracy and one way of

improving the accuracy of the proposed method is to take into account different viewpoints in order to identify the conflicting requirements [139]. Viewpoints can also identify the possible trade-offs between different requirements [139] and will pave the way for requirements negotiation.

Viewpoints has a promise to accommodate the diversity through the identification of various views, their perspectives and concerns. However, the approach can also promote scalability in the analysis through localizing the negotiation, elicitation and conflicts relative to a given perspective.

References

[1] Cloudharmony. http://cloudharmony.com/. 78

[2] Do SLAs really matter? A 1 year case study of 38 cloud services. http://blog.cloudharmony.com/2011/01/do-slas-really-matter-1-year-case-study.html. 95

[3] Forbes. http://www.forbes.com/sites/louiscolumbus/2013/09/04/predicting-enterprise-cloud-computing-growth/. 3

[4] Salah R. Agha, Mohammed H. Jarbo, and Said J. Matr. A multi-criteria multi-stakeholder industrial projects prioritization in gaza strip. *Arabian Journal for Science and Engineering*, 38(5):1217–1227, 2013. 114, 149

[5] Carina Alves and Anthony Finkelstein. Challenges in COTS decision-making: A goal-driven requirements engineering perspective. In *The 14th International Conference on Software Engineering and Knowledge Engineering*, 2002. 21, 63, 64, 67

[6] Carina Alves and Anthony Finkelstein. Negotiating requirements for COTS-based systems. 2002. 21, 63, 64

[7] Carina Alves and Anthony Finkelstein. Investigating conflicts in COTS decision-making. *International Journal of Software Engineering and Knowledge Engineering*, 13(05):473–493, 2003. 21, 63, 64, 148

[8] Carina Alves, Xavier Franch, Juan P. Carvallo, and Anthony Finkelstein. Using goals and quality models to support the matching analysis during COTS selection. *Lecture Notes in Computer Science*, Volume 3412:146–156, 2005. 21, 43, 63, 64

[9] Carina Frota Alves. *Managing Mismatches in COTS-Based Development*. PhD thesis, Department of Computer Science, University College London, 2005. 30, 50, 67, 106, 148

[10] Amazon. Amazon web services customer agreement. *http://aws.amazon.com/agreement/*. 6, 21, 27, 71, 81

[11] Amazon. Amazon EC2 service level agreement. http://aws.amazon.com/ec2/sla/, June 2013. 6

[12] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002. 46

[13] Annie I. Antón. Goal-based requirements analysis. In *The Second IEEE International Conference on Requirements Engineering,Colorado,USA*, 1996. 42, 46, 47, 58, 90

[14] Annie I. Antón. *Goal identification and refinement in the specification of software-based information systems*. PhD thesis, Georgia Institute of Technology, 1997. xiii, 39, 46, 48, 51, 82

[15] Annie I. Antón, Ryan A. Carter, Aldo Dagnino, John H. Dempster, and Devon F. Siege. Deriving goals from a use-case based requirements specification. *Requirements Engineering*, 6(1):63–73, 2001. 25, 43

[16] Annie I Antón, W Michael McCracken, and Colin Potts. Goal decomposition and scenario analysis in business process reengineering. In *Advanced Information Systems Engineering*, pages 94–104. Springer, 1994. 44

[17] Annie I. Antón and Colin Potts. The use of goals to surface requirements for evolving systems. In *The 20th International Conference on Software Engineering*, 1998. 45, 58, 61

[18] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010. 2

[19] Mohsen Asadi, Ebrahim Bagheri, Dragan Gasevic, and Marek Hatala. Goal-oriented requirements and feature modeling for software product line engineering. In *The 26th ACM Symposium on Applied Computing (SAC 2011)*, 2011. 42

[20] Yudistira Asnar, Paolo Giorgini, and John Mylopoulos. Goal-driven risk assessment in requirements engineering. *Requirements Engineering*, 16(2):101–116, 2011. 58, 60

[21] Paolo Avesani, Cinzia Bazzanella, Anna Perini, and Angelo Susi. Facing scalability issues in requirements prioritization with machine learning techniques. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 297–305, Aug 2005. 147, 154

[22] Lee Badger, Robert Bohn, Shilong Chu, Mike Hogan, Fang Liu, Viktor Kaufmann, Jian Mao, John Messina, Kevin Mills, Annie Sokol, et al. US government cloud computing technology roadmap. *NIST Special Publication*, pages 500–293, 2011. 27

[23] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001. 34

[24] Laszlo A. Belady and Meir M. Lehman. A model of large program development. *IBM Systems journal*, 15(3):225–252, 1976. 22

[25] Stephen Biggs and Stilianos Vidalis. Cloud computing: The impact on digital forensic investigations. In *the International Conference for Internet Technology and Secured Transactions*, 2009. 2

[26] Barry Boehm. Software engineering economics, 1981. 74

[27] Barry Boehm, Prasanta Bose, Ellis Horowitz, and Ming June Lee. Software requirements negotiation and renegotiation aids: A theory-w based spiral approach. In *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pages 243–243, April 1995. 53

[28] Barry W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988. 33, 34

[29] Barry W. Boehm and Tom DeMarco. Software risk management. *IEEE software*, 14(3):17–19, 1997. 10, 97

[30] Jan Bosch. *Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. 25

[31] Ronald J. Brachman and Hector J. Levesque. *Readings in knowledge representation*. Morgan Kaufmann Publishers Inc., 1985. 56

[32] Nathalie Brender and Iliya Markov. Risk perception and risk management in cloud computing: Results from a case study of swiss companies. *International journal of information management*, 33(5):726–733, 2013. 37

[33] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Sytems*, 8:203–236, 2004. 63

[34] Greg Brown, Betty H. C., Heather Goldsby, and Zhang Ji. Goal-oriented specification of adaptation requirements engineering in adaptive system. In *Proceedings of the 2006 International Workshop on Self-adaptation and Self-managing Systems*, 2006. 44

[35] David Bush and Anthony Finkelstein. Requirements stability assessment using scenarios. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 23–32, Sept 2003. 45

[36] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *High Performance Computing Simulation, 2009. HPCS '09. International Conference on*, pages 1–11, June 2009. 66

[37] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities. In *The 10th IEEE International Conference on High Performance Computing and Communication*, 2008. 2

[38] Antoine Cailliau and Axel van Lamsweerde. Assessing requirements-related risks through probabilistic goals and obstacles. *Requirements Engineering, Springer*, 18, Issue 2:129–146, 2013. 58, 59, 60

[39] Daniele Catteddu. Cloud computing: Benefits, risks and recommendations for information security. In *Web Application Security*, volume 72 of *Communications in Computer and Information Science*, pages 17–17. Springer Berlin Heidelberg, 2010. 28

[40] Kishore Channabasavaiah, Kerrie Holley, and Edward Tuggle. Migrating to a service-oriented architecture. *IBM DeveloperWorks*, 16, 2003. 20

[41] Betty H. C. Cheng and Joanne M. Atlee. Research directions in requirements engineering. In *2007 Future of Software Engineering*, FOSE '07, pages 285–303, Washington, DC, USA, 2007. IEEE Computer Society. 23

[42] Betty H. C. Cheng, Pete Sawyer, Nelly Bencomo, and Jon Whittle. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In *Model Driven Engineering Languages and Systems*, volume 5795 of *Lecture Notes in Computer Science*, pages 468–483. Springer Berlin Heidelberg, 2009. 62

[43] Ching-Hsue Cheng, Kuo-Lung Yang, and Chia-Lung Hwang. Evaluating attack helicopters by AHP based on linguistic variable weight. *European Journal of Operational Research*, 116(2):423–435, 1999. 148

[44] Lawrence Chung and Kendra Cooper. A knowledge-based COTS-aware requirements engineering approach. In *The 14th International Conference on Software Engineering and Knowledge Engineering*, 2002. 63, 64

[45] Lawrence Chung, Tom Hill, Owolabi Legunsen, Zhenzhou Sun, Adip Dsouza, and Sam Supakkul. A goal-oriented simulation approach for obtaining good private cloud-based system architectures. *Journal of Systems and Software*, 86(9):2242–2262, 2013. 64, 66, 68, 136, 137, 162

[46] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishing, 2000. 45, 49, 54

[47] Rance Cleaveland and Scott A. Smolka. Strategic directions in concurrency research. *ACM Computing Surveys (CSUR)*, 28(4):607–625, December 1996. 149

[48] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993. 42, 44, 45, 49, 55

[49] Robert Darimont and Axel van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *Proceedings of the 4th ACM SIGSOFT Symposium on Foundations of Software Engineering*, SIGSOFT '96, pages 179–190, New York, NY, USA, 1996. ACM. 49

[50] L. Davis, J. Payton, and R. Gamble. Toward identifying the impact of COTS evolution on integrated systems. In *2nd Workshop on Successful COTS*. Citeseer, 2000. 22

[51] Leticia Duboc, Emmanuel Letier, and David S. Rosenblum. Systematic elaboration of scalability requirements through goal-obstacle analysis. *Software Engineering, IEEE Transactions on*, 39(1):119–140, Jan 2013. 63

[52] Leticia Duboc, Emmanuel Letier, David S. Rosenblum, and Tony Wicks. A case study in eliciting scalability requirements. In *International Requirements Engineering, 2008. RE '08. 16th IEEE*, pages 247–252, Sept 2008. 64

[53] Leticia Duboc, David S. Rosenblum, and Emmanuel Letier. Death, taxes, & scalability. *Software, IEEE*, 27(4):20–21, July 2010. 64

[54] Eric Dubois, Michel Petit, and Eric Yu. From early to late formal requirements: A process-control case study. In *Proceedings of the 9th Internationa Workshop on Software Specification and Design*. IEEE Computer Society, 1998. 45

[55] Steve Easterbrook. *Elicitation of requirements from multiple perspectives*. PhD thesis, Citeseer, 1991. 52

[56] Steve Easterbrook. Domain modelling with hierarchies of alternative viewpoints. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 65–72, 1993. 50

[57] Steve Easterbrook. Resolving requirements conflicts with computer-supported negotiation. *Requirements engineering: social and technical issues*, 1:41–65, 1994. 52

[58] Alexander Egyed and Barry Boehm. Comparing software system requirements negotiation patterns. *Systems Engineering*, 2(1):1–14, 1999. 53

[59] Aniko Ekart and S.Z. Nemeth. Stability analysis of tree structured decision functions. *European Journal of Operational Research*, 160:676–695, 2005. 121

[60] Golnaz Elahi and Eric Yu. A goal oriented approach for modeling and analyzing security trade-offs. In *Conceptual Modeling - ER 2007*, volume 4801 of *Lecture Notes in Computer Science*, pages 375–390. Springer Berlin Heidelberg, 2007. 67

[61] John Mylopoulos Eric S.K. Yu. *Entity-Relationship Approach - ER '94 Business Modelling and Re-Engineering, Lecture Notes in Computer Science*, chapter From E-R to "A-R" - Modelling Strategic Actor Relationships for Business Process Reengineering, pages 548–565. Springer Berlin Heidelberg, 1994. 54

[62] Funmilade Faniyi, Rami Bahsoon, Andy Evans, and Rick Kazman. Evaluating security of architectures in unpredictable environments: A case for cloud. In *the 9th Working IEEE/IFIP Conference on Software Architecture*, 2011. 84, 95, 116, 117

[63] Anthony C.W. Finkelstein, Dov Gabbay, Anthony Hunter, Jeff Kramer, and Bashar Nuseibeh. Inconsistency handling in multiperspective specifications. *Software Engineering, IEEE Transactions on*, 20(8):569–578, 1994. 33

[64] Josep Oriol Fitó, Mario Macías Lloret, Jordi Guitart Fernández, et al. Toward business-driven risk management for cloud computing. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 238–241, Oct 2010. 36

[65] Jeff Frooman. Stakeholder influence strategies. *Academy of management review*, 24(2):191–205, 1999. 151

[66] Barbara Gaudenzi and Antonio Borghesi. Managing risks in the supply chain using the ahp method. *The International Journal of Logistics Management*, 17(1):114–136, 2006. 111

[67] Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. A requirements management framework for privacy compliance. In *Proc. of the 10th Workshop on Requirements Engineering (WER'07)*, pages 149–159. 62

[68] Sepideh Ghanavati, Daniel Amyot, and Liam Peyton. Compliance analysis based on a goal-oriented requirement language evaluation methodology. In *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, pages 133–142, Aug 2009. 62

[69] Martin Glinz and R.J. Wieringa. Guest editors' introduction: Stakeholders in requirements engineering. *Software, IEEE*, 24(2):18–20, March 2007. 151

[70] Michael W. Godfrey and Qiang Tu. Evolution in open source software: a case study. In *Software Maintenance, 2000. Proceedings. International Conference on*, pages 131–142, 2000. 22

[71] Derek Gottfrid. Protrated supercomputing fun! The New York Times, November 2007. 2

[72] Daniel Gross and Eric Yu. Evolving system architecture to meet changing business goals: an agent and goal-oriented approach. In *The Fifth IEEE International Symposium on Requirements Engineering*, 2001. 61

[73] Daniel Gross and Eric Yu. From non-functional requirements to design thorugh patterns. *Requirements Engineering, Springer*, 6:18–36, 2001. 54

[74] Irfan Gul and M. Hussain. Distributed cloud intrusion detection model. *International Journal of Advanced Science and Technology*, 34:71–82, 2011. 98

[75] Patrick T. Harker and Luis G. Vargas. The theory of ratio scale estimation: Saaty's analytic hierarchy process. *Management Science*, 33(11):1383–1403, 1987. 146, 154

[76] Mark Harman, Phil McMinn, Jerffeson Teixeira De Souza, and Shin Yoo. Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical software engineering and verification*, pages 1–59. Springer, 2012. 163

[77] Peter Haumer, Klaus Pohl, and Klaus Weidenhaupt. Requirements elicitation and validation with real world scenes. *Software Engineering, IEEE Transactions on*, 24(12):1036–1054, Dec 1998. 45

[78] Jim Highsmith and Alistair Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001. 34

[79] Hooman Hoodat and Hassan Rashidi. Classification and analysis of risks in software engineering. *World Academy of Science, Engineering and Technology*, 56:446–452, 2009. 145

[80] Shi-Ming Huang, I-Chu Chang, Shing-Han Li, and Ming-Tong Lin. Assessing risk in erp projects: identify and prioritize the factors. *Industrial management & data systems*, 104(8):681–688, 2004. 111

[81] Anthony Hunter and Bashar Nuseibeh. Managing inconsistent specifications: Reasoning, analysis, and action. *ACM Transactions on Software Engineering and Methodology*, 7(4):335–367, October 1998. 52

[82] IBM. IBM perspective on cloud computing, 2008. 2

[83] Waheed Iqbal, Matthew N. Dailey, David Carrera, and Paul Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871–879, 2011. 22

[84] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document, 1990. 41

[85] Joachim Karlsson, Stefan Oisson, and Kevin Ryan. Improved practical support for large-scale requirements prioritising. *Requirements Engineering, Springer-Verlag London*, 2:51–60, 1997. 113

[86] Joachim Karlsson, Claes Wohlin, and Bjrn Regnell. An evaluation of methods for prioritizing software requirements. *Information and Software Technology, Elsevier Science*, 39:939–947, 1998. 110, 113, 147, 154, 157, 161

[87] Evangelia Kavakli, Pericles Loucopoulos, and Despina Filippidou. Using scenarios to systematically support goal-directed elaboration for information system requirements. In *Engineering of Computer-Based Systems,1996. Proceedings., IEEE Symposium and Workshop on*, pages 308–314, Mar 1996. 44

[88] Rick Kazman, Mark Klein, and Paul Clements. Evaluating software architectures-methods and case studies, 2001. 95

[89] Ali Khajeh-Hosseini, David Greenwood, and Ian Sommerville. Cloud migration: A case study of migrating an enterprise it system to IaaS. In *the 3rd International Conference on Cloud Computing*, 2010. 66, 127

[90] Barbara Kitchenham. DESMET: A method for evaluating software engineering methods and tools technical report tr96-09. *Department of Computer Science, University of Keele, Staffordshire*, 1996. 148

[91] Barbara Kitchenham, Stephen Linkman, and David Law. DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal*, 8(3):120–126, 1997. 126, 127, 148

[92] Irina Todoran Koitz and Martin Glinz. Stakecloud tool: From cloud consumers' search queries to new service requirements. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 286–287, Aug 2015. 24

[93] Gerald Kotonya and Ian Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18, 1996. 24, 30, 164

[94] Jeff Kramer and Jeff Magee. Self-managed systems: an architectural challenge. In *Future of Software Engineering, 2007. FOSE '07*, pages 259–268, May 2007. 61

[95] Phillip A. Laplante, Jia Zhang, and Jeffrey Voas. What's in a name? distinguishing between saas and soa. *IT Professional*, 10(3):46–50, May 2008. 20

[96] Emmanuel Letier. *Reasoning about agents in goal-oriented requirements engineering*. PhD thesis, Université catholique de Louvain, 2001. 51

[97] Emmanuel Letier, David Stefan, and Earl T. Barr. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 883–894, New York, NY, USA, 2014. ACM. 60, 61, 163, 164

[98] Emmanuel Letier and Axel van Lamsweerde. Agent-based tactics for goal-oriented requirements elaboration. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 83–93, New York, NY, USA, 2002. ACM. 50

[99] Emmanuel Letier and Axel van Lamsweerde. Deriving operational software specifications from system goals. In *Proceedings of the 10th ACM SIGSOFT Symposium on Foundations of Software Engineering*, SIGSOFT '02/FSE-10, pages 119–128, New York, NY, USA, 2002. ACM. 49, 50

[100] Emmanuel Letier and Axel van Lamsweerde. High assurance requires goal orientation. In *Proceedings of the international workshop requirements for high assurance system, Essen, Germany*, 2002. 51

[101] Emmanuel Letier and Axel van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering*, SIGSOFT '04/FSE-12, pages 53–62, New York, NY, USA, 2004. ACM. 50

[102] Sotirios Liaskos, Rina Jalman, and Jorge Aranda. On eliciting contribution measures in goal models. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 221–230, Sept 2012. 147

[103] Sotirios Liaskos, Alexei Lapouchnian, Yiqiao Wang, Yijun Yu, and Steve Easterbrook. Configuring common personal software: a requirements-driven approach. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 9–18, Aug 2005. 46

[104] Soo Ling Lim. *Social networks and collaborative filtering for large-scale requirements elicitation.* PhD thesis, University of New South Wales, 2011. 127, 152

[105] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. Stakenet: Using social networks to analyse the stakeholders of large-scale software projects. In *32nd ACM/IEEE International Conference on Software Engineering*, 2010. 119, 147, 149, 151, 155

[106] John D.C. Little and Stephen C. Graves. Little's law. In *Building Intuition*, pages 81–100. Springer, 2008. 136

[107] Lin Liu, Eric Yu, and John Mylopoulos. Security and privacy requirements analysis within a social setting. In *11th IEEE International Requirements Engineering Conference*, 2003. 55, 62

[108] Mitch Lubars, Colin Potts, and Charlie Richter. A review of the state of the practice in requirements modeling. In *In Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 2–14. IEEE Computer Society Press, 1993. 110

[109] Leszek Maciaszek and Bruce Lee Liong. *Practical Software Engineering: An Interactive Case-Study Approach to Information Systems Development.* Pearson Addison Wesley, 2004. 126

[110] Michael Mattess, Christian Vecchiola, Saurabh Kumar Garg, and Rajkumar Buyya. Cloud bursting: Managing peak loads by leasing public cloud services. *Cloud Computing: Methodology, Systems, and Applications, CRC Press, USA*, 2011. 101

[111] Sourav Mazumder. SOA: a perspective on implementation risks. *SETLabs Briefings*, 2006. 21

[112] Frank J. Carmone Jr., Ali Kara, and Stelios H. Zanakis. A Monte Carlo investigation of incomplete pairwise comparison matrices in AHP. *European Journal of Operational Research*, 102:538„1□7553, 1997. 113

[113] Peter Mell and Tim Grance. The NIST definition of cloud computing. 2011. 24

[114] Micheal Miller. *Cloud Computing: Web Based Applications That Change the Way You Work and Collaborate Online*. Que Publishing, 2008. 2

[115] Mirko Morandini, Loris Penserini, and Anna Perini. Towards goal-oriented development of self-adaptive systems. In *Proceedings of the 2008 International Workshop on Software Engineering for Adaptive and Self-managing Systems*, SEAMS '08, pages 9–16, New York, NY, USA, 2008. ACM. 62

[116] Jean-Henry Morin, Jocelyn Aubert, and Benjamin Gateau. Towards cloud computing SLA risk management: Issues and challenges. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 5509–5514, Jan 2012. 36

[117] Haralambos Mouratidis, Shareeful Islam, Christos Kalloniatis, and Stefanos Gritzalis. A framework to support selection of cloud providers based on security and privacy requirements. *Journal of Systems and Software*, 86(9):2276–2293, 2013. 66

[118] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using non-funtional requirements: A process oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992. 43, 45, 49, 54, 75

[119] Vivek Nallur and Rami Bahsoon. Design of a market-based mechanism for quality attributes tradeoff of services in the cloud. In *The 2010 ACM Symposium on Applied Computing*, 2010. 66

[120] Vivek Nallur, Rami Bahsoon, and Xin Yao. Self-optimizing architecture for ensuring quality attributes in the cloud. In *The 2009 IEEE/IFIP WICSA/ECSA*, 2009. 66, 100

[121] K. Nigim, N. Munier, and J. Green. Pre-feasibility MCDM tools to aid communities in prioritizing local viable renewable energy sources. *Renewable energy*, 29(11):1775–1791, 2004. 157

[122] Nils J. Nilsson. Problem-solving methods in artificial intelligence. *McGraw-Hill, New York*, 1971. 43

[123] Samia Nisar and Rami Bahsoon. An economics-driven approach for automated sla negotiation for cloud services adoption: Aspoc2. In *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, pages 243–246, Dec 2013. 65

[124] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000. 24, 29, 31, 33, 74

[125] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *Software Engineering, IEEE Transactions on*, 20(10):760–773, Oct 1994. 52

[126] Colin Potts. Using schematic scenarios to understand user needs. In *The 1st Conference*

on Designing Interactive Systems: Processes, Methods and Techniques, 1995. 47, 58, 81, 90

[127] Balasubramaniam Ramesh and Vasant Dhar. Supporting systems development by capturing deliberations during requirements engineering. *Software Engineering, IEEE Transactions on*, 18(6):498–510, Jun 1992. 42

[128] William N. Robinson. Integrating multiple specifications using domain goals. In *Proceedings of the 5th international workshop on Software specification and design*, IWSSD '89, pages 219–226, New York, NY, USA, 1989. ACM. 43

[129] Colette Rolland, Carine Souveyet, and Camille Ben. Achour. Guiding goal modeling using scenarios. *Software Engineering, IEEE Transactions on*, 24(12):1055–1071, Dec 1998. 45

[130] Thomas L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15:234„1¤7281, 1977. 111, 157

[131] Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. Mcgraw-Hill, 1980. 111, 112, 117, 120, 132

[132] Thomas L. Saaty. Making and validating complex decisions with the AHP/ANP. *Journal of Systems Science and Systems Engineering*, 14(1):1–36, 2005. 146

[133] Nuno Santos, Krishna P Gummadi, and Rodrigo Rodrigues. Towards trusted cloud computing. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, pages 3–3. San Diego, California, 2009. 98

[134] Helen Sharp, Anthony Finkelstein, and Galal Galal. Stakeholder identification in the requirements engineering process. In *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*, pages 387–391, 1999. 31

[135] Darius Silingas and Rimantas Butleris. UML-intensive framework for modeling software requirements. In *Proceedings of International Conference on Information Technologies (IT*, pages 334–342, 2008. 40

[136] Stéphane S Somé. Supporting use case based requirements engineering. *Information and Software Technology*, 48(1):43–58, 2006. 40

[137] Ian Sommerville. *Software Engineering (7th Edition)*. Pearson Addison Wesley, 2004. 34

[138] Ian Sommerville and Gerald Kotonya. *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998. xv, 24, 29, 30, 32

[139] Ian Sommerville and Pete Sawyer. Viewpoints: principles, problems and a practical approach to requirements engineering. *Annals of Software Engineering*, 3(1):101–130, 1997. 164, 165

[140] OMG Unified Modeling Languages Specification. Version 1.3. *Washington, DC (Feb 1998). 4Zimmerman, P., and Symington, S., The IEEE Standardization Process, HLA*, 1998. 40

[141] Subashini Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11, 2011. 129

[142] Sam Supakkul and Lawrence Chung. Integrating FRs and NFRs: A use case and goal driven approach. *framework*, 6:7, 2005. 41

[143] Alistair G Sutcliffe, Neil A.M. Maiden, Shailey Minocha, and Darrel Manuel. Supporting scenario-based requirements engineering. *Software Engineering, IEEE Transactions on*, 24(12):1072–1088, Dec 1998. 47

[144] William Swartout and Robert Balzer. On the inevitable interwining of specification and implementation. *Communications of the ACM*, 25(7):438–440, July 1982. 45

[145] Irina Todoran. Stakecloud: Stakeholder requirements communication and resource identification in the cloud. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 353–356, Sept 2012. 65, 68, 149

[146] Irina Todoran and Martin Glinz. Towards bridging the communication gap between consumers and providers in the cloud. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*, WICSA/ECSA '12, pages 78–79, New York, NY, USA, 2012. ACM. 65

[147] Irina Todoran and Martin Glinz. Quest for requirements: Scrutinizing advanced search queries for cloud services with fuzzy galois lattices. In *Services (SERVICES), 2014 IEEE World Congress on*, pages 234–241, June 2014. 67, 68

[148] Irina Todoran, Norbert Seyff, and Martin Glinz. How cloud providers elicit consumer requirements: An exploratory study of nineteen companies. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 105–114, July 2013. 66, 68

[149] Van Tran, Jacky Keung, Anna Liu, and Alan Fekete. Application migration to cloud: A taxonomy of critical factors. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, SECLOUD '11, pages 22–28, New York, NY, USA, 2011. ACM. 128

[150] Wei-Tek Tsai. Service-oriented system engineering: a new paradigm. In *Service-oriented system engineering, 2005. sose 2005. IEEE International Workshop*, pages 3–6. IEEE, 2005. 20

[151] Axel van Lamsweerde. Divergent views in goal-driven requirements engineering. In *Joint Proceedings of the Sigsoft '96 Workshops – Specifications '96, ACM*, pages 252–256. Press, 1996. 50

[152] Axel van Lamsweerde. Requirements engineering in the year 00: A research perspective.

In *Proceedings of the 22Nd International Conference on Software Engineering*, ICSE '00, pages 5–19, New York, NY, USA, 2000. ACM. 42, 48, 51

[153] Axel van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *The 5th IEEE International Symposium on Requirements Engineering, Toronto*, 2001. 42, 43, 44, 48, 50, 51, 143, 145, 148, 152

[154] Axel van Lamsweerde. From system goals to software architecture. In *Formal Methods for Software Architectures*, volume 2804 of *Lecture Notes in Computer Science*, pages 25–43. Springer Berlin Heidelberg, 2003. 61

[155] Axel van Lamsweerde. Goal-oriented requirements engineering: From system objectives to uml models to precise software specifications. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 744–745, Washington, DC, USA, 2003. IEEE Computer Society. 45

[156] Axel van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *26th International Conference on Software Engineering*, 2004. 55, 62

[157] Axel van Lamsweerde. Goal-oriented requirements engineering: A roundtrip from research to practice. In *12th IEEE International Requirements Engineering Conference.*, 2004. 46, 53

[158] Axel van Lamsweerde. Risk-driven engineering of requirements for dependable systems. *Engineering Dependable Software Systems*, 34:207, 2013. 60

[159] Axel van Lamsweerde, Simon Brohez, Renaud De Landtsheer, and David Janssens. From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering. *Proc. of RHAS*, 3:49–56, 2003. 60, 62

[160] Axel van Lamsweerde, Robert Darimont, and Emmanuel Letier. Managing conflicts in goal-driven requirements engineering. *Software Engineering, IEEE Transactions on*, 24(11):908–926, 1998. 39, 50, 52, 59

[161] Axel van Lamsweerde, Robert Darimont, and Philippe Massonet. Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 194–203, Mar 1995. 45, 46, 48, 64

[162] Axel van Lamsweerde and Emmanuel Leiter. Integrating obstacles in goal-driven requirements engineering. In *20th International Conference on Software Engineering, Kyoto*, 1998. 58, 106

[163] Axel van Lamsweerde and Emmanuel Leiter. Handling obstacles in goal-oriented requirements engineering. *IEEE Transactions on Software Engineering*, 26:978–1005, 2000. 40, 50, 52, 53, 56, 57, 58, 59, 60, 81, 90, 115, 152, 153

[164] Axel van Lamsweerde and Emmanuel Leiter. From object orientation to goal orientation:

A paradigm shift for requirements engineering. In *Radical Innovations of Software and Systems Engineering in the Future, Springer-Verlag, LNCS*, volume 2941, pages 325–240, 2004. 55, 90

[165] Axel van Lamsweerde and Laurent Willemet. Inferring declarative requirements specifications from operational scenarios. *Software Engineering, IEEE Transactions on*, 24(12):1089–1114, Dec 1998. 47

[166] Yi Wei and M. Brian Blake. Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, 14(6):72–75, 2010. 20

[167] Linda Westfall. Software risk management. In *Annual Quality Congress Proceedings-American Society for Quality Control*, pages 32–39. ASQ; 1999, 2000. 111

[168] Wei-Wen Wu, Lawrence W. Lan, and Yu-Ting Lee. Exploring decisive factors affecting an organization's saas adoption: A case study. *International Journal of Information Management*, 31(6):556–563, 2011. 128, 130

[169] John Yen and W Amos Tiao. A systematic tradeoff analysis for conflicting imprecise requirements. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 87–96. IEEE, 1997. 32

[170] Robert Yin. Case study research: Design and methods. beverly hills, 1994. 126

[171] Shin Yoo, Mark Harman, Paolo Tonella, and Angelo Susi. Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge. In *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, ISSTA '09, pages 201–212, New York, NY, USA, 2009. ACM. 147, 153

[172] Eric Yu and John Mylopoulos. Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, pages 15–22, 1998. 40, 51

[173] Eric S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Third IEEE International Symposium on Requirements Engineering*, 1997. 45, 54

[174] Eric .S.K. Yu. Modeling organizations for information systems requirements engineering. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 34–41, Jan 1993. 51

[175] Yijun Yu, Yiqiao Wang, John Mylopoulos, Sotirios Liaskos, Alexei Lapouchnian, and Julio Cesar Sampaio do Prado Leite. Reverse engineering goal models from legacy code. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 363–372, Aug 2005. 46

[176] Kaizhi Yue. What does it mean to say that a specification is complete? In *Proc. IWSSD-4,*

Fourth International Workshop on Software Specification and Design, 1987. 43, 52, 58, 148

[177] Shehnila Zardari and Rami Bahsoon. Cloud adoption: A goal-oriented requirements engineering approach. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, SECLOUD '11, pages 29–35, New York, NY, USA, 2011. ACM. 8, 18, 24, 25, 26, 40, 45, 64, 65, 66, 67, 68, 88, 148, 152

[178] Shehnila Zardari, Rami Bahsoon, and Aniko Ekárt. Cloud adoption: Prioritizing obstacles and obstacles resolution tactics using ahp. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 1013–1020, New York, NY, USA, 2014. ACM. 37, 64, 65, 154, 157

[179] Shehnila Zardari, Funmilade Faniyi, and Rami Bahsoon. Using obstacles for systematically modeling, analysing, and mitigating risks in cloud adoption. *Aligning Enterprise System and Software Architectures, IGI Global*, pages 275–296, 2012. 18, 24, 37, 40, 59, 64, 65, 68, 115, 138, 144, 153

[180] Pamela Zave. Classification of research efforts in requirements engineering. *ACM Computing Surveys*, 29:315–321, 1997. 24, 29, 30, 43

[181] Pamela Zave and Michael Jackson. Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.*, 6(1):1–30, January 1997. 43, 44

[182] Xuan Zhang, Nattapong Wuwong, Hao Li, and Xuejie Zhang. Information security risk management framework for the cloud computing environments. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 1328–1334, June 2010. 36

[183] Yuanyuan Zhang. *Multi-Objective Search-based Requirements Selection and Optimisation*. PhD thesis, University of London, 2010. 151

[184] Didar Zowghi and Chad Coulin. Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements*, pages 19–46. Springer, 2005. 32