Elsevier Editorial System(tm) for Journal of Systems and Software
Manuscript Draft

Manuscript Number: JSS-D-12-00301R1

Title: A GOAL-ORIENTED SIMULATION APPROACH FOR OBTAINING GOOD PRIVATE CLOUD-BASED SYSTEM ARCHITECTURES

Article Type: Special Issue:FoSECloud

Keywords: Cloud Computing; System Architecture; Goal-Oriented; NFR Framework; Simulation Model; CloudSim; Little's Law; Requirements Engineering

Corresponding Author: Mr Owolabi Legunsen,

Corresponding Author's Institution: The University of Texas at Dallas

First Author: Lawrence Chung

Order of Authors: Lawrence Chung; Tom Hill; Owolabi Legunsen; Adip Dsouza; Zhenzhou Sun; Sam Supakkul

Abstract: The fast-growing Cloud Computing paradigm makes it possible to use unprecedented amounts of computing resources at lower costs, among other benefits such as fast provisioning and reliability. In designing a good architecture - the numbers, types and layouts of devices - for a cloud-based system, which meets the goals of all stakeholders, such goals need to be factored in from the earliest stages. However, there seems to be a lack of methodologies for incorporating stakeholder goals into the design process for such systems, and for assuring with higher confidence that the designs are likely to be good enough for the stated goals. In this paper, we propose a goal-oriented simulation approach for cloud-based system design whereby stakeholder goals are captured, together with such domain characteristics as workflows, and used in creating a simulation model as a proxy for the cloud-based system architecture. Simulations are then run, in an interleaving manner, against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives. We illustrate important aspects of this approach for the private cloud deployment model and report on our experiments, using a smartcard-based public transportation system.

***Cover Letter**

Dear Editor,

We have now incorporated the comments of the Reviewers in trying to improve the overall quality of our paper, as detailed in the attached Letter of Responses. Also attached, you will find two revised versions of our original Manuscript titled "A Goal-Oriented Simulation Approach for Obtaining Good Private Cloud-Based System Architectures". One is the finalized and polished version of the manuscript which is the main one we would like you to please consider. The major changes we have made to the originally submitted version are shown in-situ in the second version, which we have attached just in case it is easier for the Reviewers to use this to confirm that we have indeed addressed their concerns.

We thank you for this opportunity and look forward to you kind consideration.

Sincerely,

Owolabi Legunsen
The University of Texas at Dallas

***Highlights**

HIGHLIGHTS

- We show one way of rationally coming up with a suitable design for a private cloud.
- Stakeholder goals are identified early, analyzed and used to drive the proposed process.
- Cloud Computing simulations are used iteratively and in an interleaving manner to evaluate design decisions.
- The proposed approach borrows heavily from existing techniques in Software Engineering.

A GOAL-ORIENTED SIMULATION APPROACH FOR OBTAINING GOOD PRIVATE CLOUD-BASED SYSTEM ARCHITECTURES

Lawrence Chung[1], Tom Hill[2], Owolabi Legunsen[1], Zhenzhou Sun[1], Adip Dsouza[1], Sam Supakkul[3]

[1,2]The University of Texas at Dallas, 800 West Campbell Road, Richardson, TX 75080, USA [1]{chung, owolabi.legunsen, zxs101020, amd061000}@utdallas.edu, [2]tom.hill.fellow@gmail.com, [3]ssupakkul@ieee.org

ABSTRACT

The fast-growing Cloud Computing paradigm makes it possible to use unprecedented amounts of computing resources at lower costs, among other benefits such as fast provisioning and reliability. In designing a good architecture – the numbers, types and layouts of devices – for a cloud-based system, which meets the goals of all stakeholders, such goals need to be factored in from the earliest stages. However, there seems to be a lack of methodologies for incorporating stakeholder goals into the design process for such systems, and for assuring with higher confidence that the designs are likely to be good enough for the stated goals. In this paper, we propose a goal-oriented simulation approach for cloud-based system design whereby stakeholder goals are captured, together with such domain characteristics as workflows, and used in creating a simulation model as a proxy for the cloud-based system architecture. Simulations are then run, in an interleaving manner, against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives. We illustrate important aspects of this approach for the private cloud deployment model and report on our experiments, using a smartcard-based public transportation system.

Keywords: Cloud Computing, System Architecture, Goal-Oriented, NFR Framework, Simulation Model, CloudSim, Little's Law, Requirements Engineering

1. INTRODUCTION

Would it be possible to predict whether a cloud-based service will indeed be good enough from the perspectives of multiple stakeholder goals, such as profitability, performance and scalability? Can we do this in the early stages of development, before committing to potentially costly and time-consuming implementation and testing? Answering these questions affirmatively is important to people who want to take advantage of the many benefits, such as cost reduction, fast service provision, reliability, and the like, promised by Cloud Computing [1], [2]. In particular, designers of proposed cloud-based systems will want to investigate how to come up with designs that meet the goals of all stakeholders before it is too late, disruptive or expensive to make changes. This paper proposes one approach for doing this by using goal-orientation together with cloud computing simulations that are cheap and quick to set up, since goal-oriented techniques allow for capturing stakeholder goals and using them in exploring, analyzing and selecting among architectural design alternatives. In the proposed approach, goal orientation and simulation are used in an interleaving, iterative manner – goal-oriented aspects can be carried out simultaneously, and in any order with, simulation modeling and analysis.

By nature, early stage design in cloud-based systems is multi-stakeholder, multi-objective, multi-dimensional and large scale. It is "multi-stakeholder" in the sense that there are different types of stakeholders (e.g., cloud vendors, service providers, and end users) and "multi-objective" in that the goals of one group of stakeholders differ from those of another group and goals are oftentimes competing and conflicting, even within the same group. The "multi-dimensional" nature concerns the fact that stakeholder goals need to be

simultaneously investigated for different stakeholder groups at different levels of abstraction (hardware level, Virtual Machine (VM) level, database level, etc.). It is also "large scale" because very large numbers of individuals in each stakeholder group need to be considered while designing such systems, necessitating the use of different kinds of workloads in assessing the quality of system design. These characteristics make cloud-based system design quite challenging. Without a systematic method, such as the one proposed in this paper, it would be a daunting challenge to understand, develop, and successfully operate cloud based systems. This goal-oriented, simulation-based approach provides a fast way with little financial and manpower resources to tackle the challenge.

Users' requirements for cloud computing architecture, as well as the role of goal-oriented requirements engineering in Cloud Computing adoption have been described [3], [4]. However, there still needs to be a more systematic approach for integrating these into the architecture design process for cloud-based systems, towards reaching the level of rigor and success attained by goal-oriented techniques in traditional (not cloud-based) Software Engineering. Perhaps, the lack of such a rational approach in cloud-based system design is the main culprit in what has been described as "solutions that are looking for a problem" in the cloud [5]? Our main aim in this paper is to describe a systematic approach which may be used to design cloud-based systems based on stakeholder needs. We borrow heavily from best-of-class Goal-Oriented Software Engineering [6] techniques, while showing how such might be used in a realistic system design scenario. We also focus on the so-called private cloud deployment model, in which one entity (the Cloud Service Creator) owns and operates the datacenter but gives shared access to other entities who subscribe on a pay-as-you-go basis.

The use of simulations for investigating complex system behavior is not new. However, in the Software Engineering community, there has been a recent increase in the recognition of the role that simulations can play in the early stages of system design, especially for evaluating and validating design decisions. This is perhaps due to growing realization among researchers that rising levels of systems complexity need to be matched by more reliable ways of investigating designs in the early stages. The marriage of goal-orientation (techniques for exploration, selection among, and evaluation of architectural alternatives based on stakeholders' goals) and simulation (which is fast and easy to set up) is expected to be highly beneficial but challenging. Some work has recently emerged in this regard, for using simulation to confirm and reconfirm architectural decisions [7], for runtime monitoring and system maintenance [8], [9] and for simulating and optimizing design decisions in quantitative goal models [10]. The Cloud Computing field has also been quick to recognize the utility of simulations in investigating infrastructural level concerns [11], [12]. Our use of simulation is for investigating the impact of Cloud-Computing infrastructural design choices on the proposed system's ability to meet the needs of its stakeholders.

At a high level of description, our approach starts with the capture of stakeholder goals plus some domain characteristics such as workflows. The goals are then refined and extended quantitatively with numbers obtained from the domain, using estimation methods that we propose. All these are subsequently used to create a simulation model as a proxy for the cloud-based system architecture. Lastly, simulations are run iteratively, in an interleaving manner, against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives that have a better chance of meeting the stakeholder goals.

The main contributions of this paper are as follows:

1. A method of combining goal-oriented requirements engineering techniques with simulation for cloud computing.

2. A technique for complementing the qualitative goal models in the NFR Framework [58] with a quantitative approach.
3. The development of an interactive, iterative and interleaving simulation technique based on this quantitative approach as well as stakeholder needs.
4. A new visual notation, along with heuristics and guidelines for reasoning about the goal models in the presence of the quantitative additions.

In the rest of this paper, Section 2 provides a background to our approach, including an overview of the real world system used for illustration and some key Goal-Oriented Requirements Engineering concepts to be used in the rest of the paper. Section 3 gives step-by-step details of our proposed approach as applied to the system under study. We discuss our experimental outcomes in Section 4 and related work in Section 5. We conclude and give thoughts on future directions in Section 6.

2. BACKGROUND

2.1 Application Domain Overview: The "myki"

The "myki" [13] is a contactless smartcard system, recently created to automate and harmonize ticketing on all channels of public transport (trains, buses and trams) in Victoria, the most densely populated State in Australia. 547 million passengers boarding were reported on all modes of transportation in 2010 and the tram system in Melbourne is the largest in the world. Fig. 1 shows some essential end-user related activities within the "myki". These include adding value to smartcard (Fig. 1, Top Up), swiping card to enter vehicle (Fig. 1, Touch On), swiping card to leave vehicle (Fig. 1, Touch Off) and the automatic computation and deduction of fares. Other transactions include checking the card balance, viewing past transactions, reporting and blocking a lost card, getting periodic reports and system initiated third party payment transactions to banks. All transactions require real time communication with, and processing in, the data center. The ability of the cloud datacenter to handle such large workloads is an important question that all

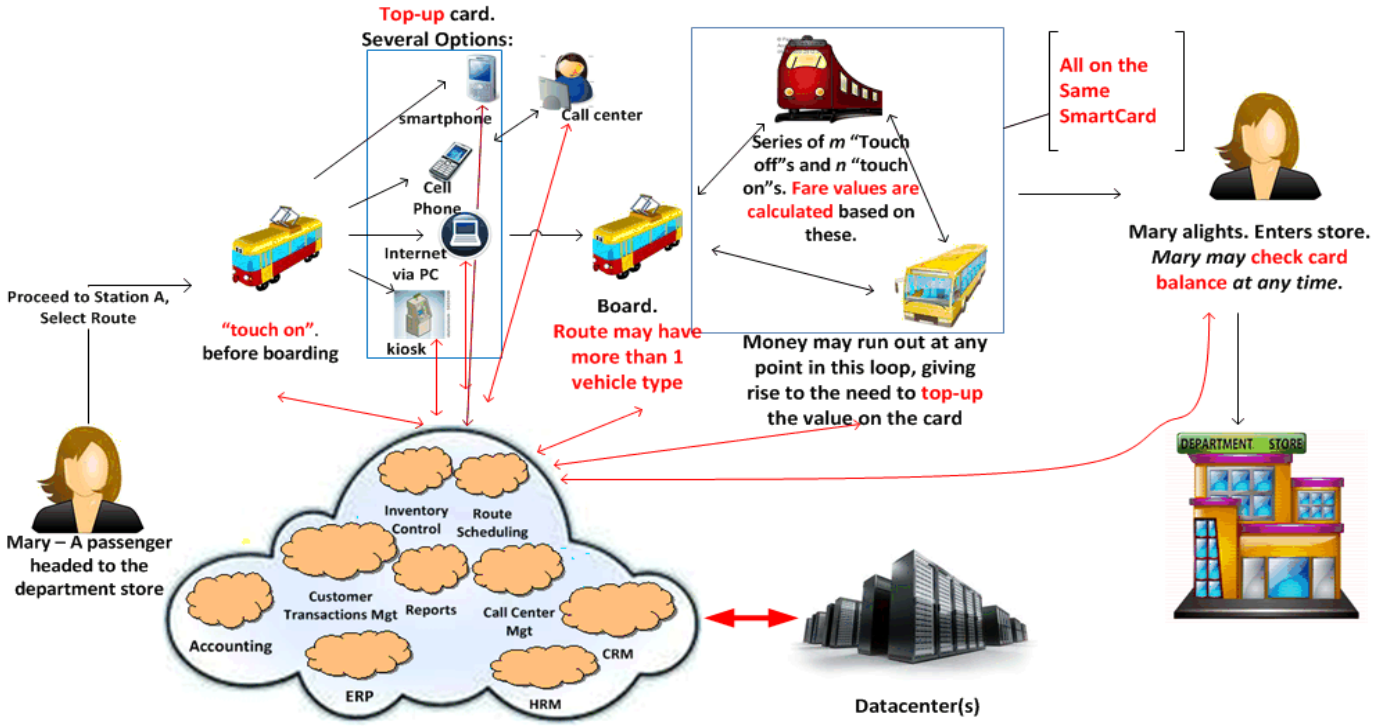


Figure 1: In the "myki", very many transactions must be processed in real time. ***How do we come up with a cloud-based design to satisfy this and other design constraints for all stakeholders?***

stakeholders will like answers to, beforehand, if possible.

The authors think that as Victoria seeks to remain one of the most livable places in the world, future improvements to the myki system are likely to become more and more important to all stakeholders. An examination of the issues associated with such a system [14] as well as projections of possible future improvements [15] might cast the "myki" as a possible candidate for migration to the cloud, in our opinion. To further motivate the use of the "myki" system as an example of how cloud-based system architecture might be designed, we consider the hypothetical cases in which (1) Melbourne earns the right to host the Summer Olympics (2) The Federal Government of Australia votes to adopt the "myki" for all public transportation in all of Australia. Technical considerations for such futuristic assumptions will likely lead architects to consider whether proposed designs will scale to the resulting increases in workload, beyond what was originally deployed. Questions about profitability and performance will also become more critical as design considerations. These goals interact and can be in conflict. For example, as end users demand higher performance levels at lower prices, the costs to service providers may increase. In this paper, we focus on such scalability, performance and profitability goals and show how our approach might be used to derive an architecture that is more likely to be good enough in the presence of multiple stakeholder groups, whose goals are also usually incompatible with one another.

2.2 Our Perspective on Cloud Computing

Assuming a virtualized datacenter, our view of what constitutes a cloud is as follows:

> *"A (private) cloud consists of a collection of virtual machines running on hosted infrastructure (including servers, processors, storage and network) owned by one stakeholder group who gives shared access to many customers in other stakeholder groups"*

In Section 2.3, we elaborate more on the important stakeholder groups in Cloud Computing but the diagram in Fig. 2 shows a side-by-side depiction of the most essential concepts in the "myki" system with those in the cloud, based on our stated perspective on cloud computing.

2.3 Stakeholders in Cloud-Based System Development

In coming up with fair system with respect to the goals of intended users, identifying the stakeholders is foundational to the development process [16], [17]. In their respective Cloud Computing Reference Architecture (CCRA) proposals, both IBM and NIST have proposed different stakeholder groups that should be identifiable in any cloud-based system development project [18], [19]. We follow the IBM perspective because it seems to be based on their experience in actually designing and building cloud-based solutions for their many clients. Moreso, if adopted by The Open Group (to which IBM has submitted their proposal), we feel that the IBM CCRA has a higher chance of becoming an industry standard. The IBM CCRA identifies the following groups of stakeholder roles:

i. Cloud Service Consumer

This is an organization, a human being or an IT system that consumes service instances delivered by a particular cloud service. The service consumer may be billed for its interactions with cloud service and the provisioned service instance(s).

ii. Cloud Service Provider

The Cloud Service Provider has the responsibility of providing cloud services to Cloud Service Consumers. A cloud service provider is defined by the ownership of a common cloud management platform (CCMP). This ownership can either be realized by running a CCMP by himself or consuming one as a service.

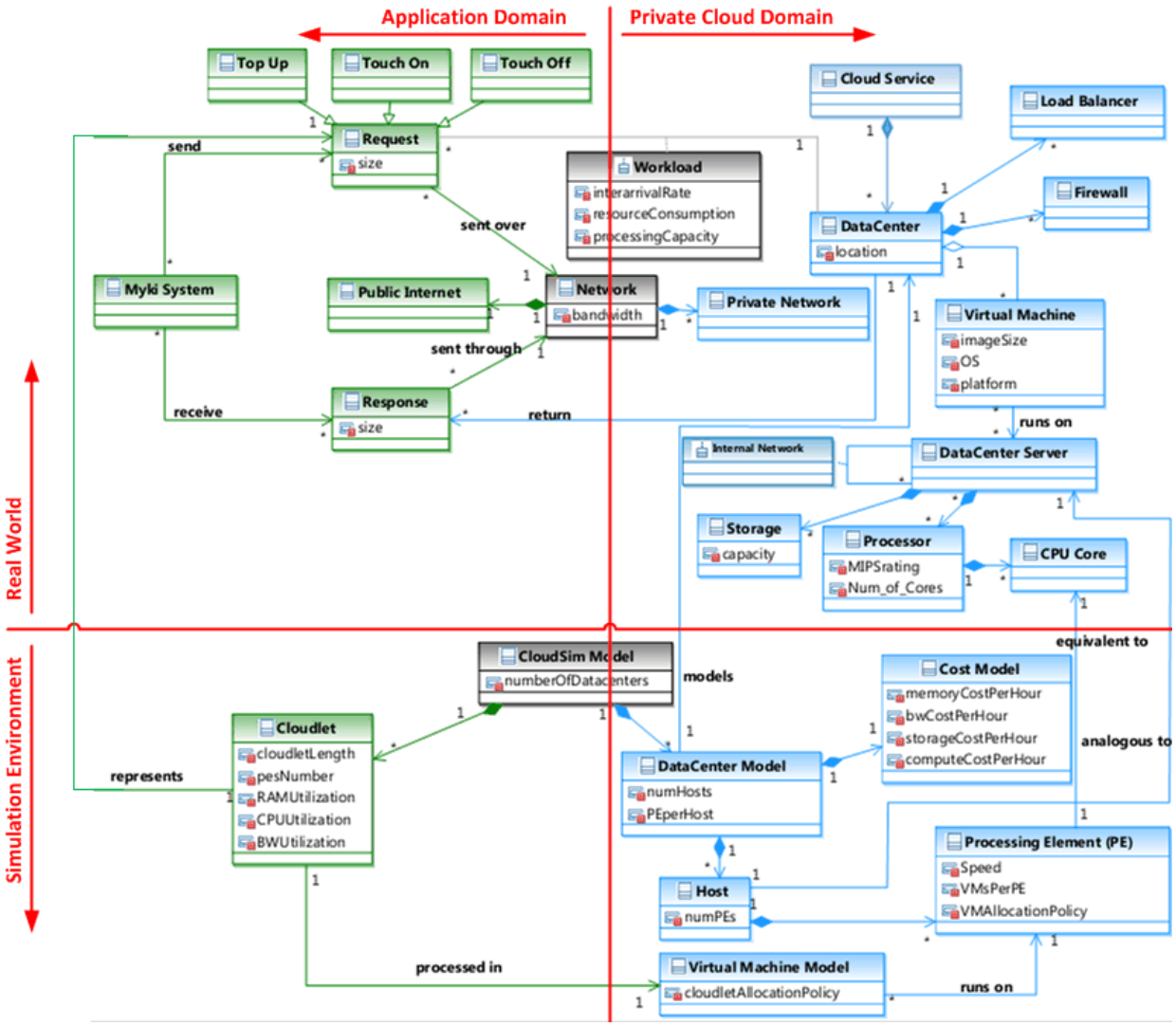


Figure 2: High-level view of the most essential concepts in the "myki" (left) and our view of a private cloud (right) for the real-world (top) and Simulation environments (bottom)

iii. Cloud Service Creator

The Cloud Service Creator is responsible for creating a cloud service, which can be run by a Cloud Service Provider or directly exposed to Cloud Service Consumers. We note that the Cloud Service Provider may comprise two distinct roles: a Service Development Architect responsible for modeling and deploying the application in the datacenter and a Service Delivery Architect responsible for running the application, mapping Virtual Machines (VMs) to hardware infrastructure and providing a cost model.

iv. End User

It is surprising that the End User is not treated as a separate stakeholder group in neither the NIST nor IBM CCRA. The reason for this omission is unstated but it seems both CCRAs assume that End User requirements are known to the Cloud Service Consumer who then uses these as a basis of selecting the services in the first place. We depart from this thinking and treat the End User as a separate stakeholder group, following the well-established fact from Software Engineering that the number one source of project failures is a lack of understanding of the End User's requirements [20].

Fig. 3 shows the hierarchy of these groups of stakeholders while Table 1 describes them within the "myki".

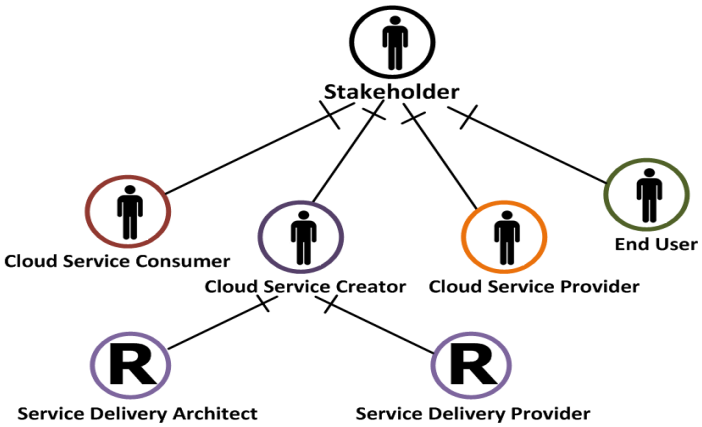


Figure 3: A Hierarchy of Stakeholders and roles in Cloud-Based System Design

2.4 Goal-oriented Requirements Engineering with the NFR Framework

A closer look at the "Goals" column in Table 1 reveals that most of the goals are expressed in subjective, non-functional terms [21]. This is typical and the problems associated with the elicitation, analysis and specification of stakeholder requirements (expressed in terms of such non-functional goals) has been studied and described in the Software Engineering literature ([6], [22] and [23]). Also, frameworks like KAOS [24], i* Framework [25] and the NFR Framework [58] have been proposed for dealing with stakeholder goals. The Softgoal Interdependency Graph (SIG) proposed as part of the NFR Framework is useful for depicting, analyzing and reasoning about softgoals - goals that are addressed not absolutely, but in a good enough sense. These kinds of goals are the focus of this work (we focus on performance, scalability and profitability for clarity and brevity). Hence, SIGs are used subsequently. A high level SIG for some goals in the myki system is shown in Fig. 4. Stakeholders are shown as agents and roles in the spirit of the i* Framework.

In SIGs, softgoals are shown as clouds with labels - a type and topic pair - beneath each cloud. Roughly, the type of a softgoal describes it while its topic (shown in square brackets) tells what it applies to. Softgoals may be refined by AND/OR decompositions. OR decompositions show design alternatives, where achieving one or more alternatives satisfices the parent softgoal. All alternatives in an AND decomposition must be achieved to satisfice the parent softgoal. The bottom leaves in a SIG should ultimately be operationalizing softgoals, represented as thick clouds and which can be actualized by an assigned agent in the domain. SIGs assist in reasoning about the interactions and conflicts among the stakeholders' goals by depicting the degree of positive or negative contributions among softgoals with pluses (+) and minuses (-). For example, the softgoal, *Profitability[cloud service]* may be "helped" by the softgoals, *Cost Effective[myki system]* and *Maximize[datacenter utilization].*

Role	Entity	Description	Goals
End User	Passengers	Patrons who need to pay fares to travel from one location to another and carry out essential activities in their daily lives	Quick and convenient request processing.
Cloud Consumer	KAMCO	They consume cloud services in order to provide services for end users to carry out their activities within the system.	Costs should not affect profit margins.
Cloud Provider	CSP	A fictional Cloud Service Provider, CSP Inc., is assumed in this paper.	Good server utilization factor. Right brokering and resource allocation policies.
Cloud Creator	HP	Owns and runs the Datacenter which is backbone of the private cloud.	Meeting SLAs agreed to with Cloud Service Providers.

Table 1: Identifying Cloud Based System Development Stakeholders in the "myki" system

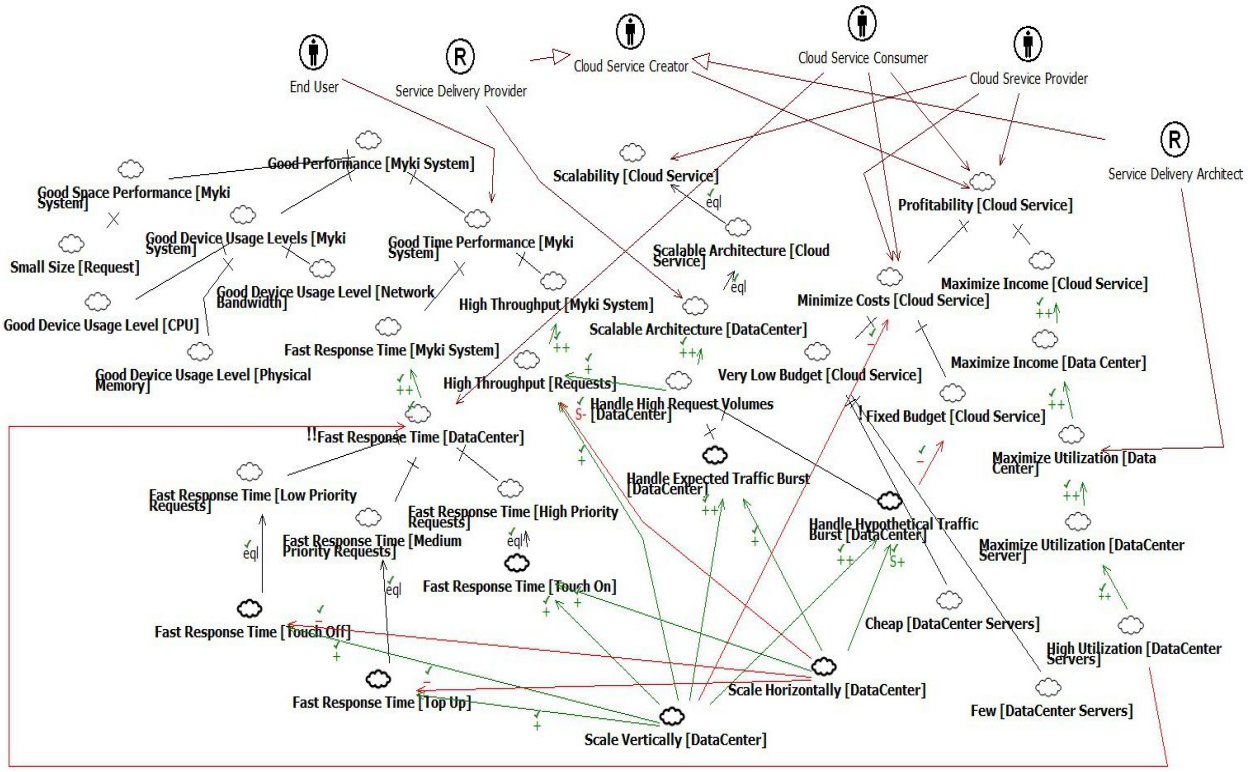


Figure 4: A decomposition of "myki" stakeholder goals using a Softgoal Interdependency Graph (SIG)

SIGs also allow for showing the level of importance of a goal relative to others in the goal model. A goal preceded by "!!" is considered very critical, one marked with "!" is critical and one that is not marked with any exclamation is neutral. As an example, Fig. 4 also shows a conflict between the (very critical) softgoal, *!!Fast Response Time[Datacenter]* and *High Utilization[DataCenter Servers]* which is needed to improve profitability. This is so because the goal, *!!Fast Response Time[Datacenter]* is very critical to end-users and, hence, the Cloud Consumer will require that the system should have very good response times. However, the goal, High *Utilization[DataCenter Servers]*, which is important to the profitability of the Cloud Service Creator, entails making such architectural decisions like using less powerful servers or adding more load to existing servers instead of buying new ones. Such decisions will save cost but are likely to have a negative impact on the *!!Fast Response Time[Datacenter]* softgoal. Hence, in Fig. 4, the goal, *High Utilization[DataCenter Servers]* (indirectly) helps the goal *Profitability[Cloud Service]* but hurts the goal *!!Fast Response Time[Datacenter]*. Similar reasoning is behind the positive and negative contributions among the other goals shown in Fig. 4. The technical challenge is how to come up with a private cloud architecture which is good enough inspite of such incompatibilities.

2.5 Using CloudSim for Cloud Based System Design

CloudSim [26] has been proposed and used as a tool for simulating and investigating cloud computing environments, based on the following premise:

> *"The core hardware infrastructure services related to the Clouds are modeled in the simulator by a Datacenter component for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on Datacenter's host components. By VM processing, we mean a set of operations related to VM life cycle: provisioning of a host to a VM, VM creation, VM destruction, and VM migration"*

In this paper, we use Cloudsim to investigate the impact of design choices made in the datacenter configuration on concerns at a higher level of abstraction, closer to stakeholders' goals. CloudSim was chosen mainly because, to the best of our knowledge, there was no other cloud simulation tool available when the research began. Other cloud simulation tools have since been proposed [27], [28] and will be investigated in the future. The two relevant CloudSim concepts with which we are concerned in this paper are:

i. Cloudlets

In CloudSim, Cloudlets may be used to represent an application or a request to an application. Cloudlets consume assigned resources and utilize them for a time calculated as a length in Million Instructions (MI) divided by the VM performance in Million Instructions per Second (MIPS)[1]. This paper uses cloudlets as single requests that arrive for processing in the datacenter.

ii. Datacenter

In CloudSim, a data center is modeled as an aggregation of computing hosts which, in turn, consist of Processing Elements (PE). A PE is analogous to a CPU core. VMs are deployed on PEs in the simulator.

In a way, a Cloudlet object in CloudSim encapsulates the left-hand side of the domain model in Fig. 2 while a Datacenter object can be made to represent the right hand side. This is so because it is the generation of requests by users in the application domain that lead to design constraints on the architecture of the system to be deployed in the cloud – essentially a virtualized datacenter according to our stated view of what constitutes (private) clouds. Hence using CloudSim for cloud-based system design involves 3 phases:

a. Obtaining relevant information like stakeholder goals, usage patterns and constraints from the application domain;
b. Mapping the information from (a.) into a simulation model in CloudSim which is then used for investigating whether various configurations and workloads can meet stakeholder goals; and
c. Mapping results from (b.) back into a system design deployment in the cloud

Section 3 describes how these phases are may be carried out for the "myki".

3. THE PROPOSED APPROACH

This approach is a 6-step process which starts with an identification of stakeholders and their goals and ends with a design that is more likely to be good enough in the presence of conflicting stakeholder goals. Additionally, application domain characteristics are obtained and translated into constraints on the system design using the CloudSim simulation model. Simulations are used to confirm and reconfirm the quality of architectural decisions at different stages of the process.

The Gane-Sarson DFD in Fig. 5 shows the 6 steps, which are not meant to be followed in strict sequence. Rather, they are envisioned for use in an interactive, interleaving and iterative manner whereby aspects of multiple steps may be carried out in parallel while revising outcomes of earlier steps where necessary, depending on new information gained from the later steps. Also, the steps represent a single experimental run that may be carried out iteratively until a good enough design is obtained. The results presented in this paper do not cover Step 6, which is part of our future work.

3.1 Step 1: Stakeholder and Goal Identification

Proper identification of stakeholders and their goals is the basis of a successful design process – one which results in a system that has a better chance of meeting the goals of all

[1] Descriptions from CloudSim Authors obtained from comments at http://groups.google.com/group/cloudsim

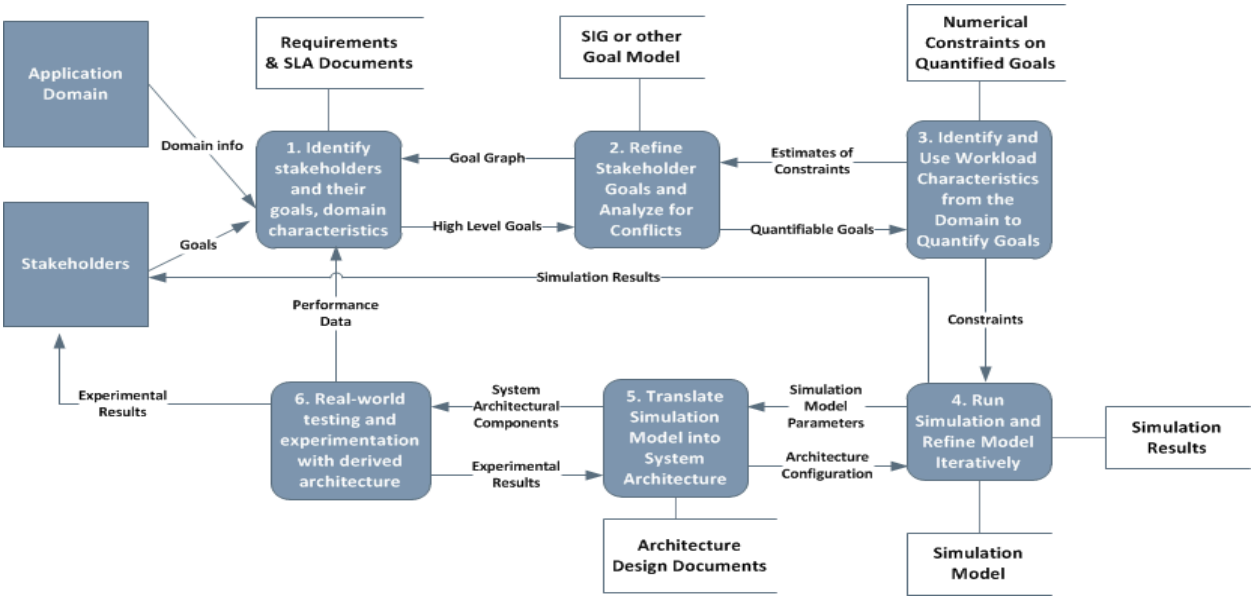


Figure 5: A 6-step process for combining goal-orientation and simulation in cloud-based system design

its intended users. Typically, the goals of stakeholders will be discovered, analyzed and refined during the requirements elicitation phase of the project. Techniques for such goal-oriented elicitation for cloud-based system development have been proposed [3], [4]. Table 1 summarizes the results of such elicitation for the 4 stakeholder groups discussed in Section 2.3, for the "myki".

3.2 Step 2: Analyzing Stakeholder Goals for Conflict Identification

Having identified the stakeholder goals, the next step is to represent the goals, along with the stakeholders interested in them in the form of a SIG diagram. Other frameworks can also be used but we have selected the SIG for reasons specified in Section 2.4. Fig. 4 shows one such representation wherein stakeholder(s) interested in a particular goal, as well as any specific roles that they play in the system, are shown alongside the high-level goal model. In Fig. 4, stakeholders are represented as agents (circled stick symbol) and roles (circled R) in the spirit of the i* Framework. The resulting diagram is henceforth referred to as an Agent-SIG. Subsequently, the Agent-SIGs are simpler than Fig. 4 for clarity in presenting the interaction between stakeholders, their goals, simulation results and the interrelationships between these. In practice, even more model refinement would be required, whereby each high level goal is sufficiently AND/OR decomposed until specific operationalization, assignable to an agent in the domain is derived.

3.3 Step 3: Quantitatively Augmenting the Qualitative Goal Model with Domain Specific Information

In order to use simulation, which is quantitative, to evaluate the impact of design decisions on the degree of satisficing of softgoals, it is necessary to augment the qualitative goal model (Agent-SIG) from Step 2, shown in Fig. 4, with numbers that guide decision making. This entails the translation of stakeholders' requirements from the application domain into numerical target values that the eventual system design is expected to meet or exceed for the related softgoals to be considered satisficed. We refer to these target values as "design constraints". Typically, these constraints will be evident, at least partially, during the requirements elicitation phase for a cloud-based development project [4] and can be of different types. In this paper, we focus on design constraints related to profitability, scalability and performance goals. This section shows how initial estimates of design

constraints on these goals may be obtained, in the "myki" example. These estimates will be used to create a baseline design which is then improved upon in subsequent iterations.

One question concerns the impact of errors in the initial estimates on the number of iterations needed for our technique to converge. We plan to experimentally investigate this in the future. However, conventional wisdom, based on experience with iterative techniques in Numerical Analysis, seems to suggest that the number of iterations in the overall process will tend to increase in direct proportion to the errors in the initial estimates. To this end, we feel that special care should therefore be taken while estimating the design constraints. Also, to start the process of estimation, one softgoal needs to be selected. We suggest the selection of a softgoal that is deemed very critical to the end-user ("customer is king") and/or a softgoal which allows the creation of a queuing or similar performance model.

It is important to point out that the aim of this step is not to replace any of the reasoning or elaboration activities and techniques normally associated with goal models. Neither is it meant to be a proposal for a new kind of quantitative goal model. Rather, it complements existing techniques in order to further facilitate the use of goal-orientation with simulation. The utility of the SIG produced in Step 2 to the activities in Step 3 includes:

1. The SIG can help stakeholders negotiate and agree on target values that define what it means for the goals in the SIG to be considered good enough.
2. It helps requirements engineers to discover what domain specific phenomena need to be quantified and fed into the simulation model.
3. It serves as the basis of reasoning about the degree to which the softgoals have been satisficed.

In the rest of this section, we show how the performance, scalability and profitability goals for the "myki" were quantitatively augmented, based on refinements shown in Fig. 4.

3.3.1 Quantifying Performance Goals

In refining the high-level performance softgoal*, Good Performance[Myki System]*, as shown in the Agent-SIG of Fig. 4*,* we adopted the classic NFR Framework definitions and approach to dealing with performance goals, referred to as the *Performance sort* [22], [54]*.* To start the estimation process, we considered the end-user goal, *!!Fast[response times]* to be very critical from the End User's perspective. Thus, this softgoal is selected initially. We note, however, that this choice is by no means unique and will vary from project to project. Next, we try to answer the question of what it might mean for the performance related softgoals in the qualitative goal model to be considered satisficed.

For the "myki" one key performance objective is for the system to process individual transactions within the smallest amount of time possible, while still processing the maximal volume of requests per day. This objective fits well into the interaction among *space performance*, *time performance* and *device utilization* as espoused in the *Performance sort* technique and captured in the Agent-SIG of Fig. 4*.* But, how do we derive these quantities like number of simultaneous transactions, maximum processing time per request, and total number of requests per day which, among others, form the key *performance variables* that are of interest to stakeholders? The maximum allowable processing time for each request will ideally be stated in the SLA agreements. For the sake of discussion in this paper, we take this value to be *12 seconds*, comparable to what is common in most POS systems [29]. For the number of requests per day, we found that the current workload on the "myki" system is 800,000 requests per day (rpd). This value, combined with other domain knowledge from the existing system is then used along with Little's Law, as shown in Section 3.3.1.1, to compute how many simultaneous requests the datacenter must be able to handle in order to meet the performance goal of *12 seconds* average response time.

Realistically, a more refined treatment might be desirable in identifying and deriving numerical values for the related performance variables. For instance, instead of computing the processing times or number of requests per day for all requests as an aggregate like we do in illustrating our ideas in this paper, these values may be computed for the individual types of request (i.e. touch on, touch off, top-up). All the estimation techniques discussed in the rest of this section will remain applicable, and the increase in the amount of computations is expected to be offset by the benefits to performance engineers in terms of a more detailed view of the system under investigation.

3.3.1.1 Estimating Performance Variables using Little's Law

Given the way that we are choosing to measure the performance goal for the "myki", as described in the previous section, we need a way to relate the different performance variables together mathematically. In our case, we found Little's Law [30] to be useful. Little's Law defines a relationship among a queuing system's average arrival rate, average processing time and average throughput. Little's Law states that, if

λ = average number of items arriving per unit time,

L = average number of items in the queuing system and

W = average waiting time in the system for an item.

Then, $L = \lambda W$ *(1)*

Because cloud-based systems will consist of requests that are eventually processed in a datacenter, with some reasonable assumptions, we think that it should be possible to model most (private) cloud-based applications as queuing systems over which Little's Law can be applied to gain intuition for creating the simulation model to be used during system design.

3.3.1.2 Some Assumptions made in using Little's Law for the "myki'

In using Little's Law to create a queuing model for the arrival of requests in the datacenter, the following assumptions were made about the system:

i. The datacenter is a single channel server.
ii. The datacenter will handle all requests uniformly.
iii. Data needed to process requests is completely and accurately contained within the datacenter.
iv. The response time for a request is equal to the processing time of each request in the datacenter, neglecting latencies in client devices and the network.
v. Multiple datacenters may conceptually be treated as a single datacenter, assuming the right task scheduling and resource allocation policies among them.

These assumptions make it possible to model the entire "myki" as a queuing system in line with the fundamental assumptions of Little's Law. In addition to these assumptions, the datacenter is initially treated as a black box into which requests flow and out of which responses come after some processing time. The internals of the datacenter are then considered as the model is improved subsequently.

3.3.1.3 Calculations and Estimates

In computing the smallest amount of time possible while still processing the maximal volume of requests per unit time, the following facts from the domain, adapted from information obtained from actual "myki" engineers, were used:

- The system currently handles 800,000 requests per day (rpd)
- 75% of requests arrive during, and are split equally between, the morning and evening rush hour periods. The remaining 25% are evenly spread over the rest of a 12-hour day.
- Rush hour is between 0700-0930 and 1700-1900 on weekdays in Melbourne.

Current average request arrival rates per day are as follows:

- *Morning rate = 300,000 requests ÷ 2.5hours = 120,000 requests per hour*

- *Evening rate = 300,000 requests ÷ 2hours = 150,000 requests per hour*
- *Non-rush hour rate = 26,667 requests per hour*

To compute the average number of requests to be simultaneously processed in the datacenter under any of these workloads, let

L = average no. of requests processed simultaneously by the datacenter;
W = average processing time per request = 1 minute = 0.01667 hour;
λ = arrival rate of requests to the datacenter = 150,000 requests/hour.

By *(1), L = λW = 150,000 requests/hour X 0.01667 hour = 2500 requests*

This implies that, to handle the same workload as the current system, the cloud-based system should, on the average, be able to handle 2500 requests simultaneously. For other design scenarios, such as the hypothetical nationwide adoption of the "myki" system, similar calculations can be used to find out how many requests the system must be able to process simultaneously. This value is then used subsequently to form input to the simulation model. Instead of randomly selecting and testing designs from the design space, the designer is, thus, handed a more precise starting point. Note that we have assumed a uniform distribution for the request inter-arrival rates for brevity. Little's Law holds regardless and other probability distribution functions may be investigated in practice.

3.3.2 Quantifying Scalability Goals

The concept of a scalability goal has recently been more precisely defined in terms of specified scaling assumptions [59], wherein a *scalability goal* is

> *"a goal whose definition and required levels of goal satisfaction make explicit reference to one or more scaling assumptions".*

In turn, a *scaling assumption* is

> *"a domain assumption specifying how certain characteristics in the application domain are expected to vary over time and across deployment environments".*

We found this definition and related treatment of scalability more suitable for our goal-oriented simulation approach, since it allows us to evaluate possible system behavior based on domain assumptions. There are other definitions of scalability as a system goal (e.g., [60]) which may be more germane to other kinds of projects.

In order to use this definition, however, the question arises as to what domain characteristic(s) the scaling assumption(s) should be based on. We restrict our focus in this paper to workload-related domain characteristics. Specifically, at the risk of appearing to use a rather crude measure of scalability, we will only use the definition of scalability as "acceptable performance under increasing workloads" [61]. It is not our aim in this paper to show the proper treatment of the individual softgoals. Rather, we try to show the impact of design decisions on the degree to which multiple softgoals are simultaneously satisficed. More detailed treatment of scalability as a softgoal may be found in the Software Engineering literature [59], [60].

We define our *scaling assumptions* in terms of four kinds of workloads outlined in Table 2. The *Current* workload refers to the amount of load currently handled by the "myki" while the *Peak* workload represents the expected load at peak capacity. The values used for the *Current* and *Peak* are somewhat obfuscated from real data obtained from "myki" operators. Next, in order to show how the cloud might be used to scale this system even further, the *Olympic* workload considers the case where the Cloud Service Consumer needs the system to handle temporarily large spikes in usage as may occur, if Melbourne were to host the Summer Olympics in the near future. Lastly, the *Australia* workload seeks to see how

Workload	Requests/day	Justification
Current	800,000	Statistics from current system
Peak	3,000,000	Expected workload at capacity
Olympics	6,000,000	Assuming that influx of tourists causes use of public transport to double
Australia	16,814,521	If Melbourne current usage is representative of national patterns, this number is proportional to the peak capacity that the current system is expected to handle.

Table 2: Different workloads to which the "myki" is expected to scale in the cloud

scalable the system might be for very large scale permanent expansion in its use, as may occur if every state in Australia decides to adopt the "myki". Both the *Olympic* and *Australia* scenarios are purely hypothetical although it should not be hard to see how they may correspond to real-world use cases. All four workload types represent the *scaling assumptions* that we have made and are by no means exhaustive.

One factor that needs to be accounted for is whether other system goals are expected to have fixed or changing target values as the workload varies. Cases where system goals are expected to remain fixed have been defined as *scalability goals with fixed objectives* (SGFO) while those in which the system goals can vary have been defined as *scalability goals with varying objectives* (SGVO) [59][2]. For the "myki", one SGFO is that the system should be able to handle all scaling assumptions while maintaining VM utilization between 40% and 75%, where utilization is computed as the fraction of time during which each VM is busy. An example of SGVO would be that some reasonable amount of performance degradation, in terms of longer response times, might be expected for the *Australia* scaling assumption.

3.3.3 Quantifying Profitability Goals

Constraints that impact on profitability can be viewed from the perspective of multiple stakeholder groups. For the "myki", passengers care about how much they are charged per transaction as well as how much taxpayer money the Government is spending to support the system. Cloud Consumers want to make enough money from fares to offset costs incurred in using the cloud service. Cloud Service Providers and Cloud Service Creators will similarly want to ensure that the cost of the services provided are offset by what they charge to their respective clients. We assume that annual cash inflow from fare collection for the *Current* workload is about $100,000,000, that revenues in other workloads are directly proportional and that fare collection is the only revenue source. Will cloud usage make the system cost-effective/profitable to all stakeholders? Will the system be able to sustain itself without Government help? This is important as it is not likely that fare prices charged to passengers will change merely because the system is migrated to the cloud platform. Also, prices charged by Cloud Service Creators (HP in the "myki" example) are typically fixed. Hence, the system design will need to ensure minimal costs. For simplicity, we consider the system cost effective and profitable for all stakeholders as long as Cloud Computing and Server costs do not exceed 5% of total revenue in the *Current, Peak* and *Olympic* Workloads and 50% in the *Australia* workload.

3.4 Step 4: Iterative Simulation and Model Refinement

The aim in this step is to translate the constraints from Step 3 (Section 3.3) into simulation models that serve as a proxy for the actual cloud-based system, through which the quality of design decisions can be investigated for impact on the satisficing of softgoals within the goal model (SIG).

3.4.1 Cloud Computing Simulation Modeling with CloudSim

In CloudSim, modeling the cloud is essentially an activity in deciding the parameters of a suitable virtualized datacenter, which is then internally converted to a discrete event simulation model For our experiments, the constraints on the system design, derived from the previous step are used to inform initial parameter selection which is then improved

[2] The abbreviations are not part of the original definitions, but have rather been used for brevity in the rest of the paper.

subsequently, based on simulation results and the degree of stakeholder goal satisfaction. Next, we briefly describe initial parameter selection, modeling and simulation model refinement steps.

3.4.1.1 Mapping from Quantified Goals to CloudSim Model

In Section 2.5, we gave a high level introduction to the two main CloudSim concepts that are needed to model a cloud-based system design: Cloudlets and Datacenters. In this Section, we show how the analysis done so far can be mapped to into a Simulation Model in CloudSim, in terms of these two entities.

3.4.1.1.1 Modeling Application Requests as Cloudlets

Table 3 describes some attributes of the Cloudlet class that were used in the simulations. The *cloudletLength* attribute represents the number of CPU instruction cycles consumed by a request. Existing techniques for estimating resource usage in virtualized environments (e.g., [31]) do not use such fine grained parameter and work mainly for existing systems. Our approach focuses on early stages of development. Hence, the challenge is to more reliably approximate this low-level, machine and implementation dependent variable before system development. We propose some techniques for approximating this value, depending only on information that can be estimated at design time:

i. If the system exists and it is possible to obtain/estimate the number of instructions currently being processed per second, the server hardware being used and the utilization of the server, then the number of instructions in a request may be approximated from:

$$\frac{\text{MIPS rating of the server}}{\text{Requests/sec at 100\% utilization}}$$

ii. If the server on which the system is being run can be determined but there is no way to obtain/estimate the requests per second, then the tpmC metric of the server may be obtained from the TPC-C benchmark [32] and used in the formula given in (i). In this case, however, the complexity of the transactions in the system under study has to be estimated as a fraction or multiple of those in the TPC-C (or similar) benchmark.

iii. If the requests per seconds can be obtained/estimated but not the MIPS rating of the actual server used, then the MIPS rating of a server with the closest semblance to the one being used may be used as an approximation

iv. If neither the MIPS rating nor the number of requests can be estimated, the problem may be narrowed down by taking a look at the architecture application and estimating the number of instructions in each component.

For the myki system, assuming HP ProLiant BL680c G7 servers running on Intel Core i7-980X processors (147,600 MIPS) process the 800,000 requests per day at the common server utilization of 15%, calculations yield 11.07 million instructions (MI) per request. Using the peak value 3,000,000 requests per day under the same assumptions yielded a value of 2.95 MI/request. We take this as a range within which the actual value may lie. Calculations using the method in (ii) above yielded a value 7.01 MI/request. Interestingly, this is within the range obtained from the first method. This gives some initial confidence

i. Attribute	Value	Meaning/Rationale
cloudletLength	12 MI	Obtained from calculations
pesNumber	1	1 CPU core of the right capacity is sufficient to process this cloudlet.
cloudletFileSize	1913 bytes	Typical file size at checkout for most e-commerce sites
utilizationModelCPU	Full	Each cloudlet will use an entire CPU core
utilizationModelRam	Full	Same meaning as in CPU
UtilizationModelBw	Full	Same meaning as in CPU

Table 3: Some Cloudlet Properties for a Single Request. Meanings of variables available in the CloudSim API online

Property	Value	Explanation
Number of Datacenters	1	
PEs per host	36	Assuming HP ProLiant BL680c G5 Servers running on the Intel Xeon X7460 processor (36 cores).
Number of Hosts	70	70 X 36 = 2520 > 2500, Assuming 1 VM per PE, 1 request per VM.
MIPS per PE	11,757 MIPS	Comparable to the MIPS of a single core in the Intel Core i7 920 (Quadcore) running at 2.66 GHz.
Storage Cost	$0.0134/GB/hr	Extrapolated from Microsoft Azure pricing.
Compute Cost	$0.04/hour	Extrapolated from Microsoft Azure pricing.
Bandwidth Cost	$0.00028/GB/hr EGRESS	Extrapolated from Microsoft Azure pricing.
Memory Costs	$0.05/MB/hr	

Table 4: Initial Datacenter Configuration. (**Host:** a server installation, **PE:** Analogous to a CPU core.)

about the estimation methods and suggests that multiple methods may be used complementarily. We conservatively use the worst case computed value of 12 MI per request in the rest of this paper.

3.4.1.1.2 Modeling the Data Center Architecture

The practice in the myki system of routing all requests through a single datacenter makes it plausible to model the initial cloud data center as being in single location. The data center configuration shown in Table 4 is the proposed initial design and derives mostly from the estimated design constraints discussed previously. 1–to-1 mapping of VM to CPU core and request to VM have been pessimistically used in coming up with the baseline and will be optimized subsequently.

3.4.2 Using Simulation to evaluate degree of Goal Satisfaction

The multidimensional nature of cloud-based system design requires that the impact of design decisions on goal satisfaction be simultaneously evaluated with respect to the stakeholders, their goals, the proposed system configuration and various workloads that the system is required to support. To better manage this iterative evaluation, we introduce the visual notation in Fig. 6 to help the designer assess impact of particular designs the stakeholder goals. Design constraints (Sections 3.3.1 to 3.3.3) inform the mapping from application domain constraints to simulation model parameters. Hence, the notation in Fig. 6 is proposed to help visualize the impact of design decisions within the simulation model on the satisfaction of design constraints, in various iterations. When combined with the Agent-SIG diagram from Fig. 4, this notation can facilitate reasoning about the various dimensions involved in cloud based-design. We illustrate the use of this notation subsequently, for the "myki" example.

The visual notation may be interpreted as follows:

i. Simulation model parameters are shown at the bottom of the diagram for different experimental runs, showing both request parameters (in green) and datacenter parameters (in blue), stacked on each other to save space. For example, the leftmost configuration (Run 1) at the bottom of Fig. 6 follows from Tables 3 and 4. Progression from the simulation model configuration in one experimental run to the next depends on design decisions based on the degree to which all stakeholders' goals have been met. We illustrate this in the "myki" design in subsequent sections.
ii. Design constraints derived from the domain are linked to the goals they impact on by dashed lines. Exact constraints that must be met in order to meet a goal (and satisfy interested stakeholders) are shown in red boxes while constraint values derived from simulation are shown in lilac boxes.
iii. Constraints may be linked to parent goals or to sub-goals.
iv. The degree of constraint satisfaction is mapped to the degree of goal satisficing in

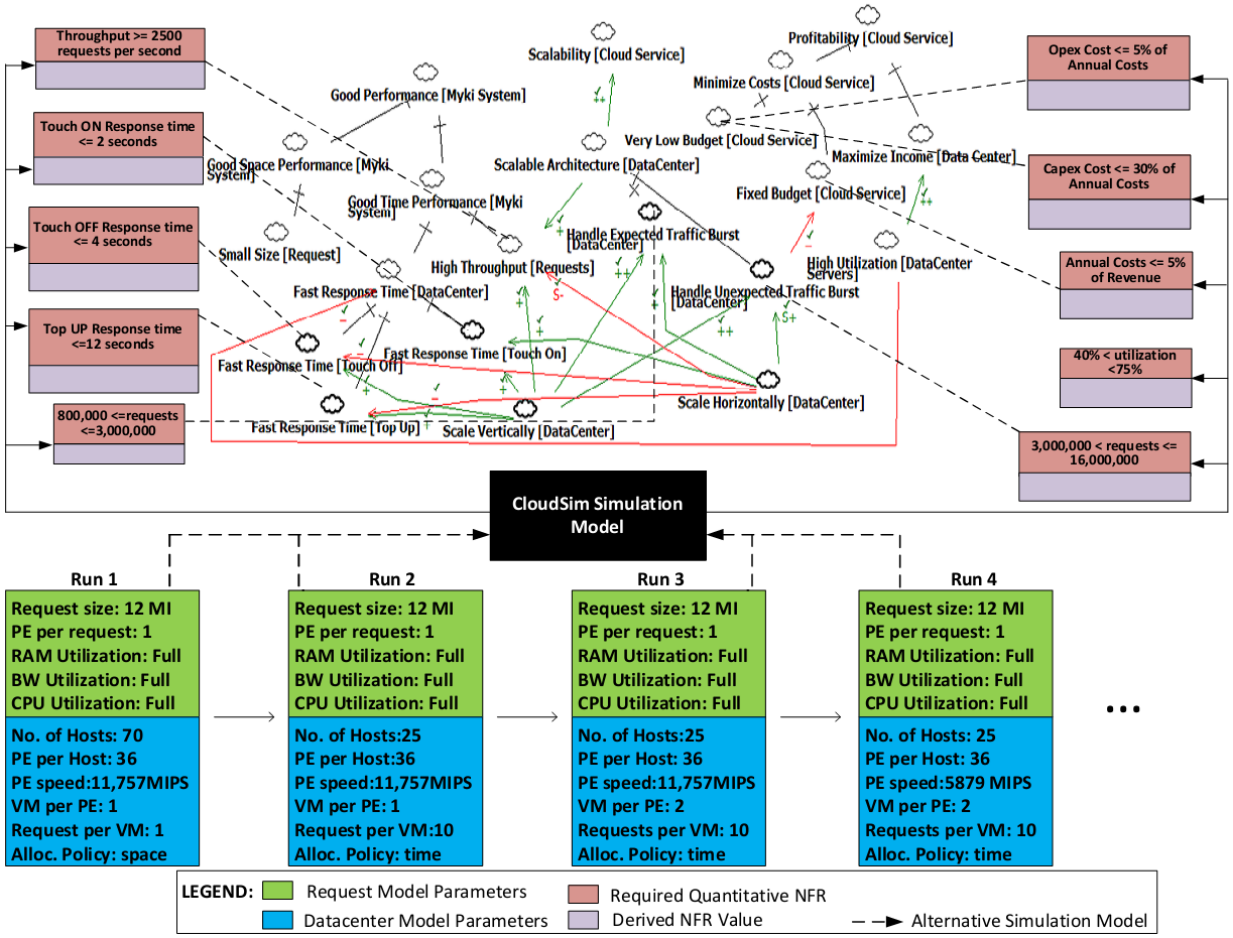


Figure 6: Visual notation for assessing impact of design decisions, represented in the simulation model, on the degree of goals satisfaction during some sample iteration steps.

the NFR Framework according to Table 6, which reads as, "If the degree of goal satisfaction is as shown in the *Constraints* row, then the associated softgoal shall be satisficed to the degree shown in the *Associated Softgoal* row for the matching case".

v. Normal rules for reasoning about SIGs remain. Quantities derived from quantitative requirements and simulation results are used as a sort of claim softgoal [58] to more properly justify why the softgoals are assigned various degrees of satisfaction.

vi. Reasoning about the interaction among the constraints, softgoals and simulation results is based on the rules shown in Table 5. For example, in the Agent-SIG of Fig. 8, the form of the "*Response Time <= 12 seconds*" (*RT12*, for short) constraint matches RULE 1 (row 1) in Table 5. Simulation results show that, for all 3 scenarios response times are are (much) less than 12 seconds. Therefore, by Case 1 of RULE 1, the *RT12* constraint is considered *satisfied*. Case 1 of Table 6 is then used to infer that the softgoal, *!!Fast Response Time[Touch On],* is satisficed, since *RT12* is linked to it. For the "*40%<utilization<70%*" (*UtilRange*, for short) constraint in Fig. 8 matches RULE 4 in Table 5. Since the utilization values obtained from simulation are much lesser than the required lower bound, by Case 2 of RULE 3, *UtilRange* is, thus, *unsatisfied*. Next, Case 3 of Table 6, the softgoal, *High Utilization[Datacenter Servers]* to which *UtilRange* is linked, is *denied.* The other associations between constraints and goals in Fig. 8 are treated similarly.

One issue concerns the implicit assumption we have made in the describing the visual

RULE	FORM		Case 1	Case 2	Case 3	Case 4	Case 5
1	var ≤ y	Result	res < y	res = y	res > y	res >> y	!def(res)
		Constraint	sat	p.sat	p.unsat	unsatisfied	unknown
2	var ≥ y	Result	res > y	res = y	res < y	res << y	!def(res)
		Constraint	sat	p.sat	p.unsat	unsatisfied	unknown
3	x≤var≤y	Result	x<res<y	res = y or res = x	res < x or res > y	res << y or res >> y	!def(res)
		Constraint	sat	p.sat	p.unsat	unsatisfied	unknown
4	var<y, var>y or x<var<y	Treat as the Rule 1,2 or 3 most closely associated, but leave out Case 2					
5	var = y	Result	res = y	res ≠ y	N/A	N/A	N/A
		Constraint	sat	unsatisfied	N/A	N/A	N/A

(Abbreviations: **var**="variable whose value is being measured", **x & *y*** = "*the (upper or lower) bound on the value of* **var**", **res** = "the actual value of *var* obtained from simulation results", **sat** = "satisfied", **p.sat** = "partially satisfied", **p.unsat** = "partially unsatisfied", **!def(res)** = "*res* is undefined")

Table 5: Reasoning about the impact of simulation results on constraint satisfaction

	Case 1	Case 2	Case 3	Case 4	Case 5
Constraint	Satisfied	Partially Satisfied	Unsatisfied	Partially Unsatisfied	Unknown
Associated Softgoal	Satisficed	Weakly Satisficed	Denied	Weakly Denied	Undecided

Table 6: Mapping levels of constraint satisfaction to the degree of softgoal satisficing

notation, viz.: constraint symbols can only link to one softgoal. What if multiple constraints link to the same goal, having different levels of satisfaction? One option is to extend normal SIG reasoning techniques to handle this situation. This is open to further research.

3.4.3 Initial Simulation, Results and Analysis

For the *Current*, *Peak* and *Olympic* workloads, using the estimated arrival rates and initial configuration discussed earlier, the first simulation experiment shows a very small, constant processing time and cost per request as shown in Table 7 and Fig. 7. For the !!Fast[response times] softgoal alone, this is a good thing. But, considering other goals, this is unlikely to lead to a fair system for all stakeholders. Annual Cloud Computing costs grow too quickly (cost as percentage of revenue is almost constant) as a 1-to-1 VM to request mapping is used (see iteration 1, Table 7 and Requests per minute and VM Usage graph, Fig. 7). The updated Agent-SIG in Fig. 8 indeed shows that multiple goals in the system have indeed been sacrificed in order to satisfice the critical performance softgoal while the Agent-SIG in Fig. 9 shows which stakeholders' goals is (un)met. Steps taken to improve this initial design are discussed next.

I	Workload	Host	VMs	RPM	Policy	p.time	p.cost	a.p.cost p	a.s.cost	t.a.cost
1	Current	70	2500	2500	space	0.14	2158.61	4,043,766.76	1,400,000.00	5,443,766.76
	Peak	261	9375	9375	space	0.14	8306.14	15,560,056.15	5,220,000.00	20,780,056.15
	Olympics	521	18750	18750	space	0.14	16612.3	31,120,112.31	10,420,000.00	41,540,112.31
2	Current	4	125	2500	DW	19.85	141.01	18,693,518.32	80,000.00	18,773,518.32
	Peak	14	469	9375	DW	19.54	738.18	97,096,234.02	280,000.00	97,376,234.02
	Olympics	27	938	18750	DW	19.54	1476.37	194,193,783.39	540,000.00	194,733,7839
3	Current	7	250	2500	time	0.60	216.84	1,701,061.25	140,000.00	1,841,061.25
	Peak	27	938	9375	space	0.10	1021.63	1,307,129.33	540,000.00	1,847,129.33
	Olympics	53	1875	18750	time	0.60	2048.60	16,070,808.36	1,060,000.00	17,130,808.36

(Abbreviations: **I**="Iteration", **RPM** = "Requests per minute", **Policy** = "Cloudlet Allocation Policy", **DW** = "Dynamic Workload Allocation Policy", **p.time** = "Processing time per request", **p.cost** = "processing cost for all RPM", **a.p.cost** = "Annual Processing Cost", **a.s.cost** = "Annual Server Cost", **t.a.cost** = "Total Annual Cost"

Table 7: Simulation results for 3 iterations over the *Current, Peak* and *Olympics* workloads for different configurations. Annual costs are extrapolated from simulation times.

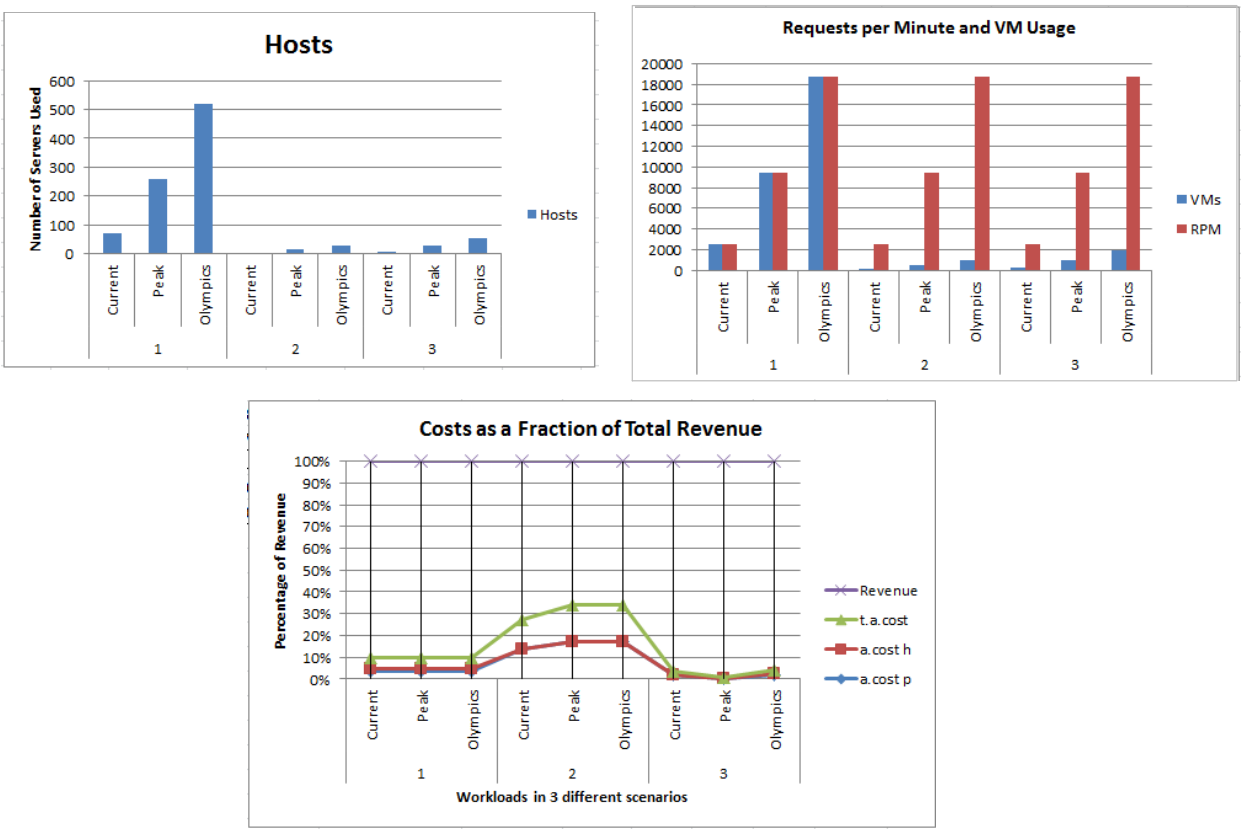


Figure 7: Graphical view of some of the results in Table 6

3.4.4 Model Refinement

Based on the simulation results in the first run, the current configuration, if deployed, is not likely to result in a fair system from the viewpoint of all stakeholders. Hence, the system design, as captured in the simulation model, will need to be improved towards meeting the goals of as many more stakeholders as possible. To do this, new architectural decisions are made based on insights from the initial results as well as the currently unmet stakeholder goals. For instance, the following decisions were made to improve the initial design for the *Current* workload:

- Assuming the 3-tiered software architecture style, the initial design for the *Current* workload suggests having 2,500 VMs, each having its own front-end, business logic and database. This is infeasible because while the front end and business logic layers may be horizontally scaled out like this, traditional database management systems do not lend themselves to such scaling out. Hence, the decisions to use one central database, have the VMs communicate with it via a load balancer (as a transaction processing (TP) monitor) and use a second database server for failover. These decisions will be reflected in the final architecture
- The baseline design used a queue length of 1. To improve utilization, let VMs handle 20 requests threads at once.
- The size of each VM is reviewed downwards and the allocation policies are changed from the space-shared policy used in the baseline to a time-shared policy for both cloudlets and VMs. These policies are defined in CloudSim [26]
- Because of the above improvements, the number of servers used in the processing of requests is reduced from 70 to 25.
- With the changes to the baseline design, the database tier is observed to be the most likely bottleneck. Database architects will have to manage this effectively.

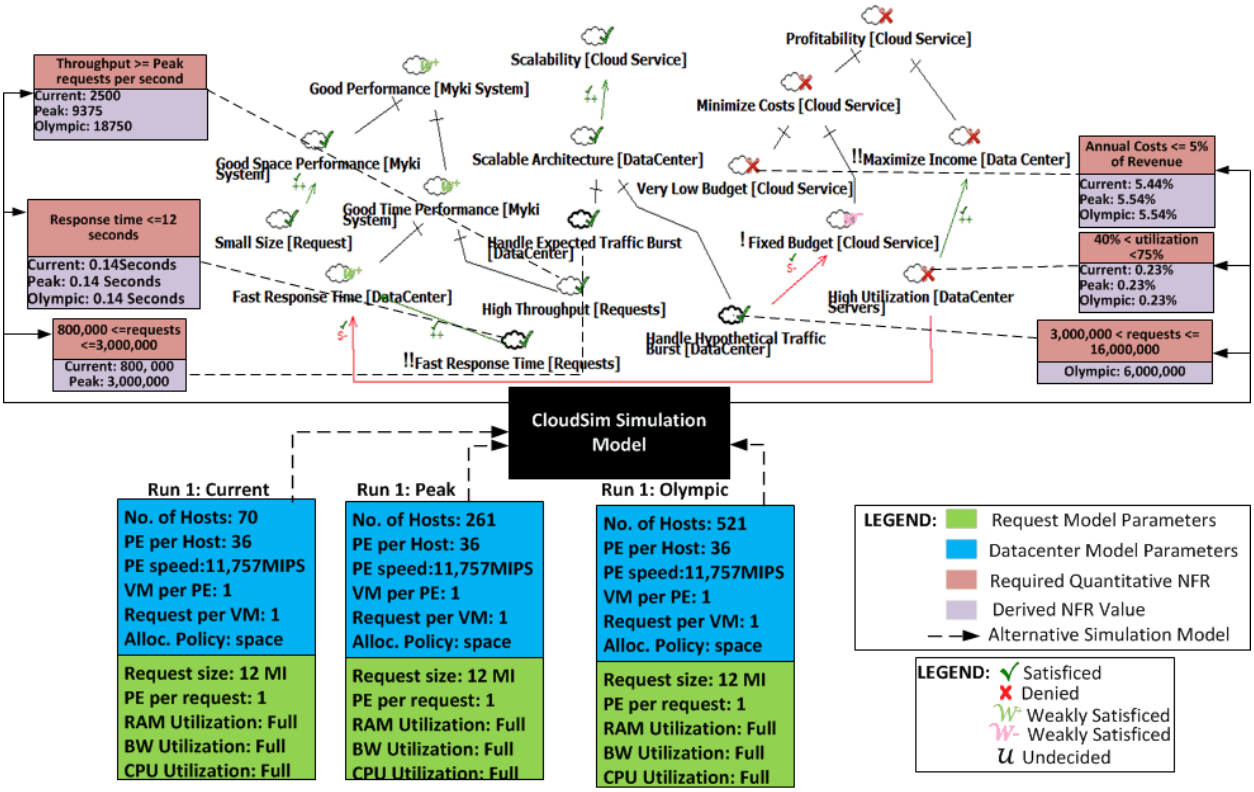


Figure 8: Updating the augmented SIG with Simulation results. Multiple goals are denied while satisficing one critical softgoal

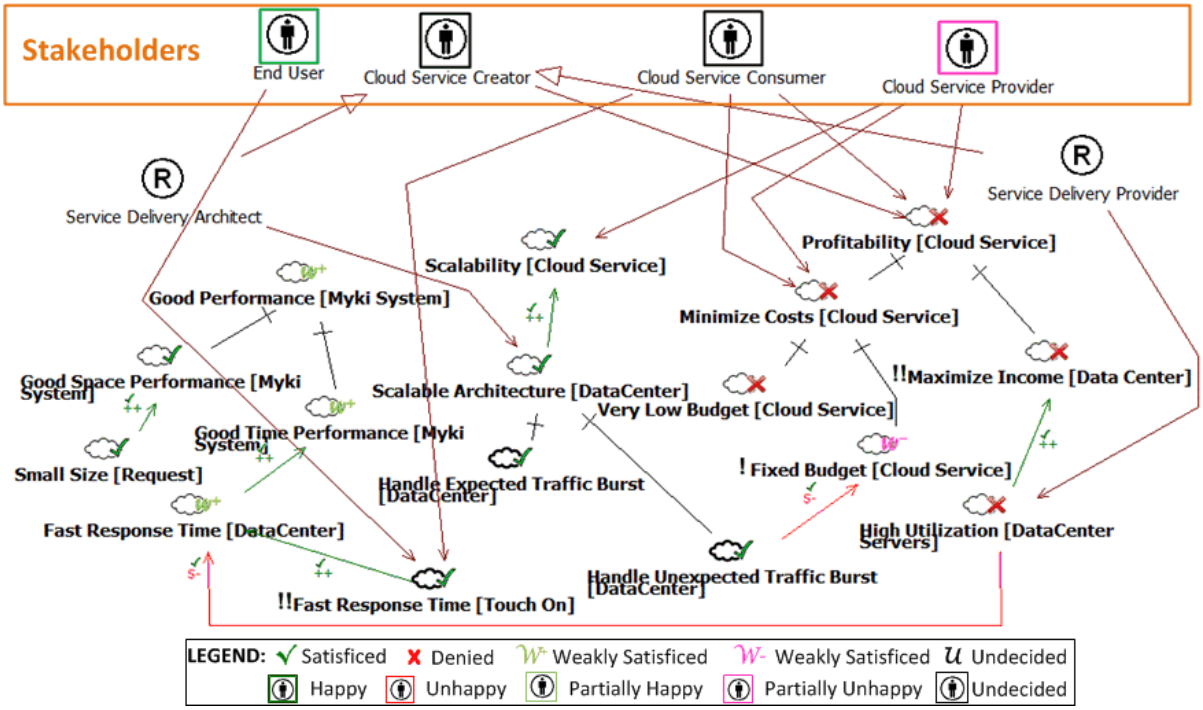


Figure 9: Updated Agent-SIG showing that meeting one critical softgoal satisfies some stakeholders but will likely result in an unfair system for other stakeholder groups.

Similar modifications are made for the *Peak* and *Olympic* workloads and the simulations are re-run. Simulations based on these improvements show some improvements in meeting other stakeholder goals. The results are shown in Fig. 7 as well as the second row in Table 7 which show that, although we use fewer VMs and Hosts (Servers) and the system is better able to scale, the response time and cost constraints have been violated. Fig. 10 shows updates to the augmented Agent-SIG after running the simulation for the "improved" configuration. Note that Fig. 10 combines the augmented SIG with the Agent-SIG diagrams for brevity.

To improve even further, we reduce the number of requests handled per VM from 20 to 10, since it seems that having greater queues consumes more computational resources and leads to the much higher response time and cost factors. We also change the request (cloudlet) allocation policy within CloudSim from *DynamicWorkload* policy to a *time-shared* policy and explore the possibility of allocating more VMs per CPU core. The results of these are shown in the third row of Table 7 and in Fig. 7. As Fig. 11 shows, these changes result in a design that is likely to better meet the stakeholder goals. Further refinements and optimizations can be carried out, if needed.

The analysis so far has evaluated whether certain configurations are good enough to meet stakeholder goals for a particular kind of workload. More specifically, with respect to the workloads described in Section 3.3.2, while the design so far may cater to the *Current*, *Peak* and *Olympic* workloads, it may not be good enough for the *Australia* workload considering

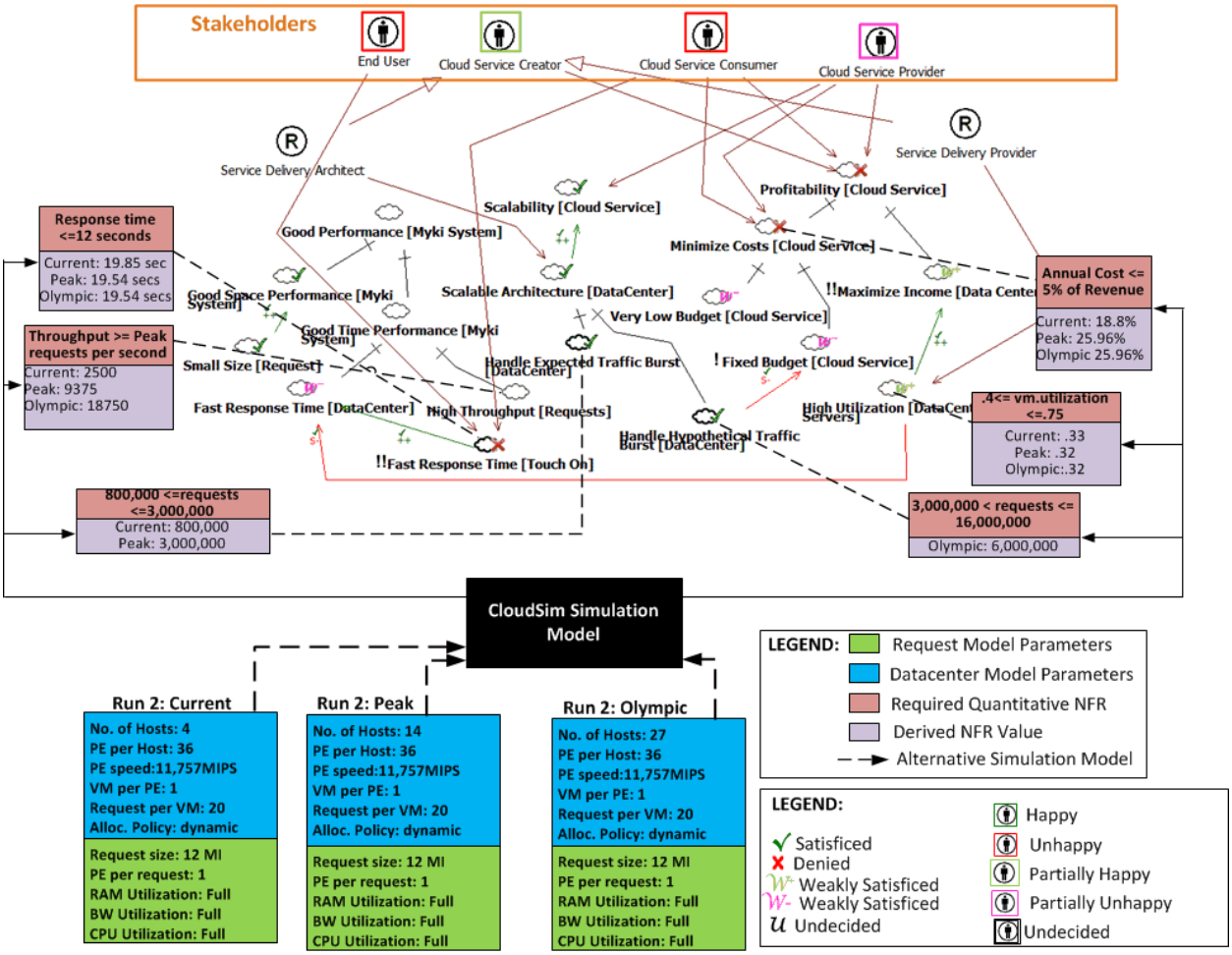


Figure 10: Improving initial design for scalability (utilization) sacrifices performance and profitability.

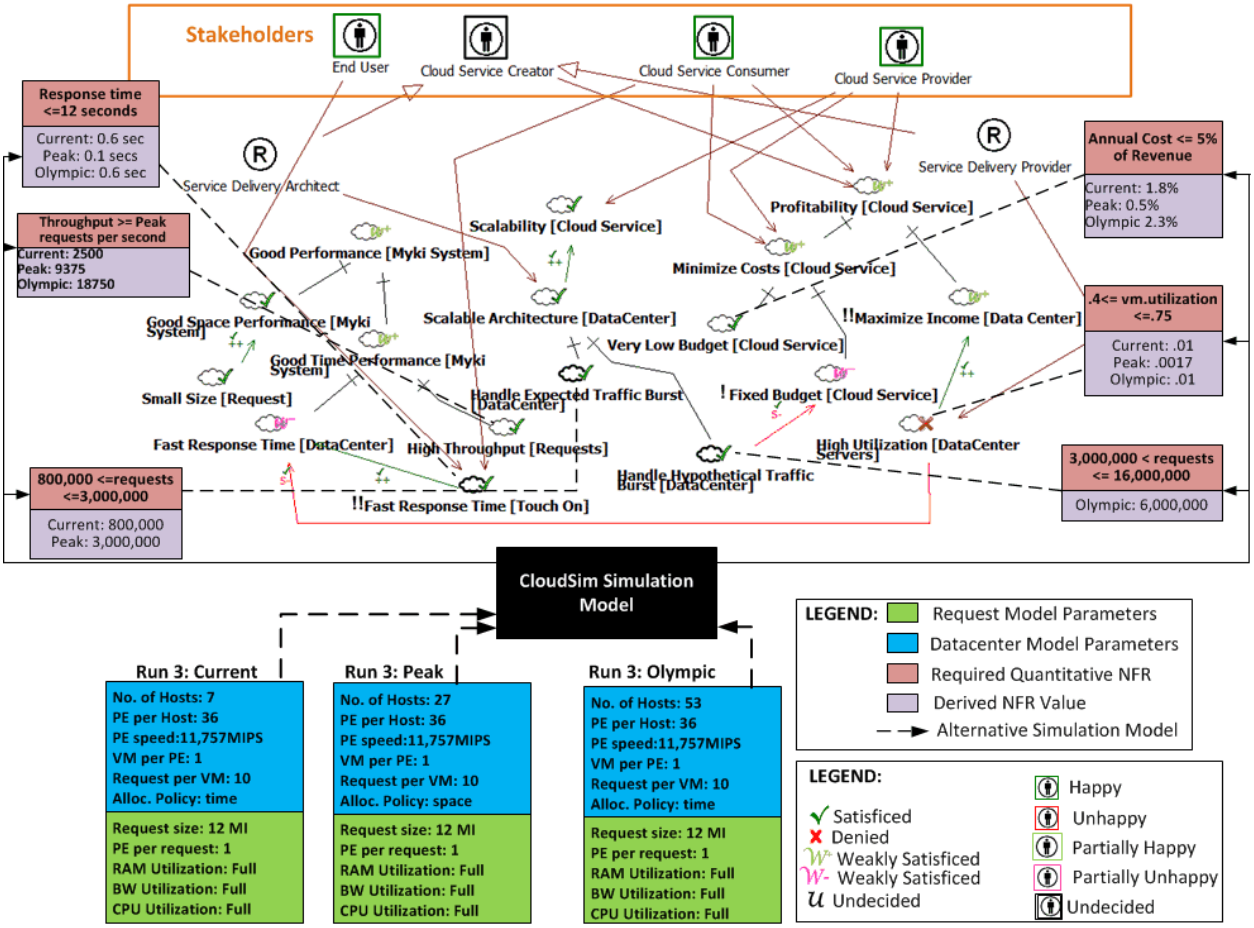


Figure 11: The configurations in the third run seems to satisfice the profitability and performance goals simultaneously, while partially satisficing for scalability (poor utilization)

its wider geographical scope and much larger scale. In the next section, we show how a good enough design for the *Olympic* workload may be derived, assuming the same constraints but with much larger budget and revenues.

3.4.5 Designing for a different workload type

The workloads considered so far shared some similar characteristics which enabled us to consider them simultaneously. In this section, we consider a workload with significantly different characteristics: the *Australia* workload. This workload results from the hypothetical case that all 6 states in Australia adopt the "myki". Designing for this workload will differ significantly from others considered so far in the following ways:

- The budget and revenue from fare collection will be about 6 times as large as the Melbourne only case (there are 6 states in Australia)
- A choice will have to be made between using a centralized datacenter like the one utilized so far or a set of datacenters distributed around Australia, since distance from datacenter is likely to lead to added performance bottlenecks
- If multiple datacenters are to be used, how many should be used and where should they be located for best performance, given the fixed cost?
- If multiple datacenters are used, network and other costs will be more significant.

Granted that many application-level and data storage related optimizations will need to be made for best performance in the *Australia* workload, we continue to use the same request

model from earlier examples to keep the discussion simple. First, we evaluate the profitability, scalability and performance tradeoff for a single datacenter configuration and see how well this meets stakeholder goals. Based on the simulations, more datacenters can be considered in the architecture. Keeping with the same assumptions and estimation technique from Sections 3.3.1.2 and 3.3.1.3, Table 8 and Fig. 12 shows the results of running the simulation for the *Australia* workload, using the earlier described visual notation.

For the second run, we randomly halved the capacity of the datacenter and bandwidth costs used in the first iteration for the *Australia* workload and used 2 such datacenters instead of 1, we observed some interesting results, as shown in Table 8 and Fig. 13. First, requests sent to the datacenter closest to them had a much less processing time (12.8 seconds) than

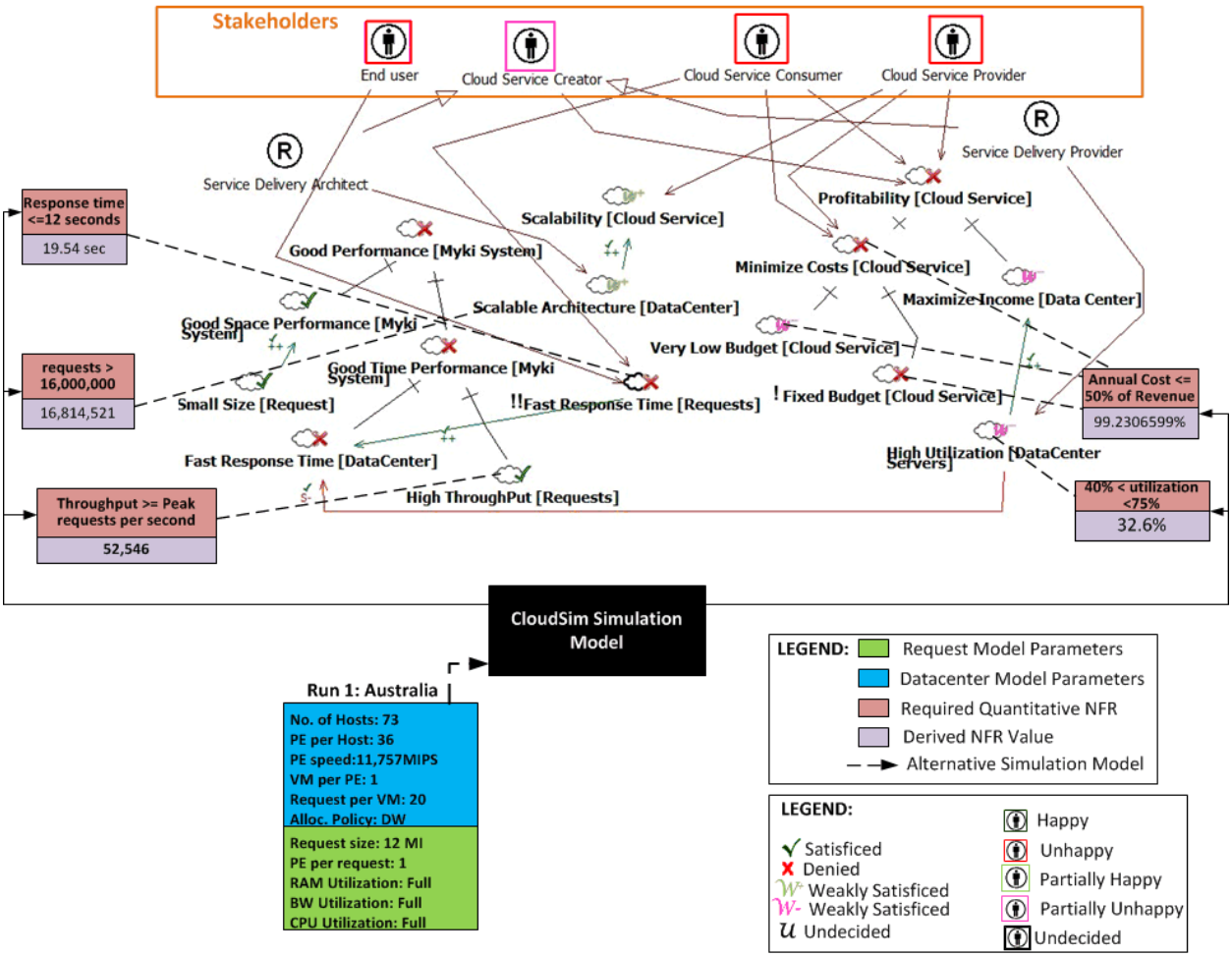


Figure 12: Even with optimizations that worked for the other workloads, using a single datacenter for the *Australia* workload does not meet most of the stakeholder requirements.

I	Hosts	VMs	RVM	p.time	a.cost p	a.cost h	t.a.cost
1	73	2628	52546	19.54	1,945,142,017.49	1,460,000.00	1,946,602,017.49
2	73	2628	52546	DC1:12.8 DC2: 31.74	1,089,811,117.77	1,460,000.00	1,091,271,117.77

(Abbreviations: **I**="Iteration", **RPM** = "Requests per minute", **Policy** = "Cloudlet Allocation Policy", **DW** = "Dynamic Workload Allocation Policy", **p.time** = "Processing time per request", **p.cost** = "processing cost for all RPM", **a.p.cost** = "Annual Processing Cost", **a.s.cost** = "Annual Server Cost", **t.a.cost** = "Total Annual Coast")

Table 8: Simulation results for the Australia Workload. Annual costs calculated from simulation times.

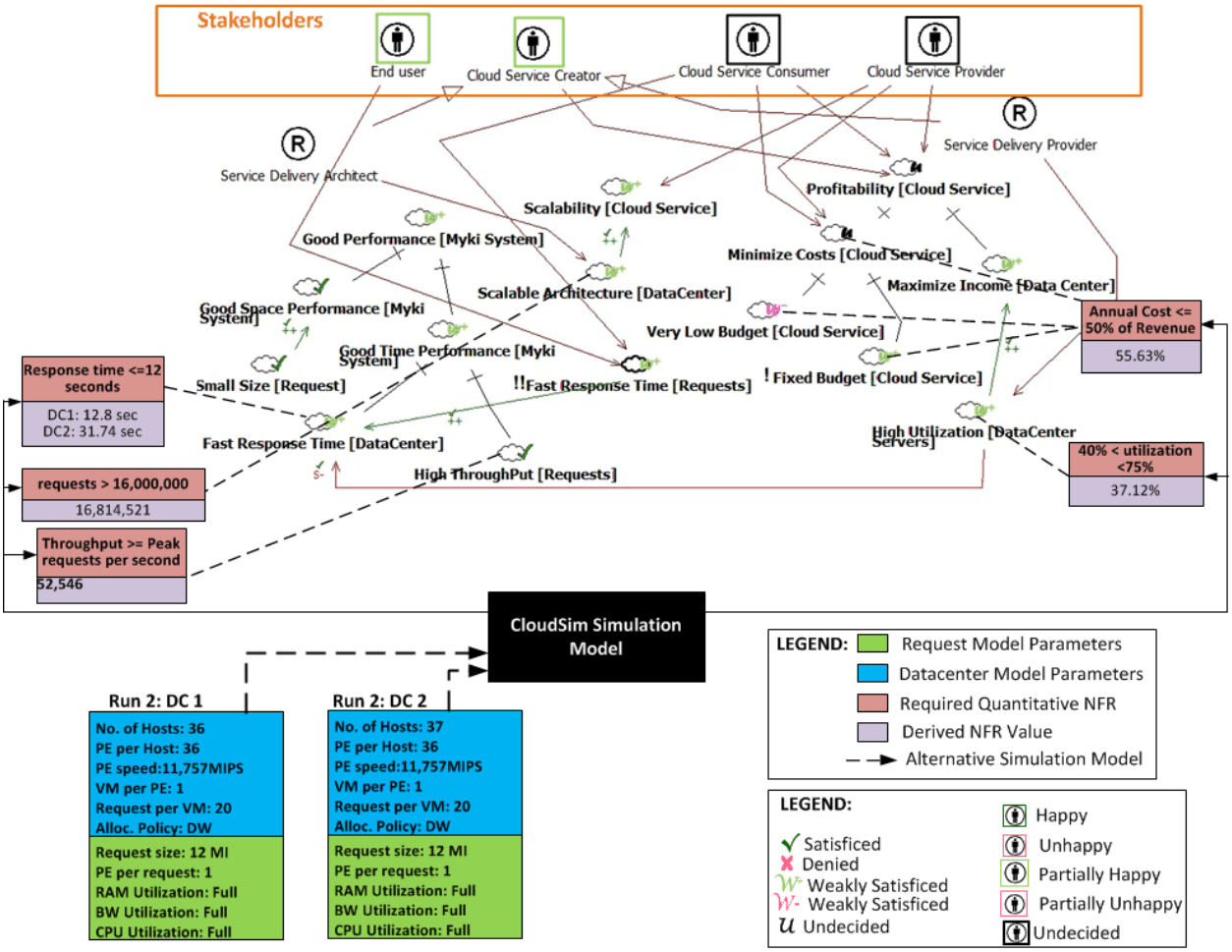


Figure 13: Using two datacenters (DC1 and DC2) shows higher chances of meeting stakeholder goals. Better selection of datacenter locations, among other things, could lead to more improved results

requests sent to the datacenter farther away (31.74 seconds). Secondly, even though the average processing time for the 2 datacenter scenario (22.27 seconds) was just a little higher than the average processing time in the 1 datacenter scenario, Cloud Computing and Server costs reduced drastically. These results suggest that, with more careful planning, locating bigger datacenters nearer to more densely populated regions reduce costs and improve performance even further, while maintaining good scalability. This seems to confirm earlier findings [34] on the impact of the distance from datacenters on cloud computing costs and performance. This issue has also been defined as a research problem in Cloud Computing, the solution to which may lead to techniques for optimizing among datacenter location, network performance, costs and other variables [35].

3.5 Step 5: Translating Derived Model into System Architecture

In this Step, results from the simulation process are translated back into private cloud architecture in the application domain. In Steps 1 and 2 (Sections 3.1 and 3.2) we discussed how to obtain information from the application domain that imposes design constraints on the cloud based system. Steps 3 and 4 (Sections 3.3 and 3.4) showed how to translate these constraints into the CloudSim simulation model which is then used, as a proxy for real system architecture, to investigate whether certain configurations will meet the goals of the system. If the configuration is not good enough, changes are made and simulations used to reconfirm their quality, until a satisficing design is obtained.

The simulations in Step 4 have the desired effect that they narrow down the design space

from infinitely many designs to a more useful subset. However, it is likely that there will still be a large number of designs in this subset and exploring them all is practically infeasible. One solution is to iteratively explore and select among incrementally better alternative designs based on the final simulation model. To do this, one starts with a few candidate designs, perhaps armed with the knowledge of creating similar systems in the cloud. These are then evaluated and the one which is most optimal with respect to the tradeoffs in the stakeholder goals is then selected and further improved. Techniques and criteria for selecting among architectural alternatives in Software Engineering have been proposed [36], [37]. In this paper, because of space constraints, we simply show one candidate design and discuss how architects may improve on it in real-world applications.

In Fig. 14 we show one such candidate design for the "myki", where the datacenter equivalents of elements from the simulation model are referred to as the "Request Processing Unit (RPU)" for convenience. The numbers and capacities of the devices in the RPU have been omitted from Fig. 14 since these vary from workload to workload. In addition to the profitability, scalability and performance goals, the Service Delivery Architect will have to optimize the datacenter architecture for other design goals. For instance, because sensitive personal (credit card) information are critical considerations, other likely bottlenecks in the real-time processing of such large volume of requests are (i) the transactional nature of current RDBMS implementations (which makes scalability harder to achieve) and (ii) the fact that third party servers will need to be contacted (e.g. for processing credit card payments) over which the designers of the "myki" have no control. These and other constraints have been identified [38] but cannot be directly simulated in CloudSim. Hence, in the real world, the architect will have to account for these and other goals in the system design while still satisfying the system's constraints and meeting the goals of all stakeholders. This is one reason why some devices like firewalls and management servers which were not considered in the simulation model appear as part of the system architecture shown in Fig. 14.

3.6 Step 6: Testing and Real World Experimentation

In the previous five steps, particular care has been taken to consider as many factors as possible. Despite this, mapping from designs obtained from the ideal world of the simulator into real test beds may not be exact. Some reasons for such divergence of the simulation model from reality include non-functional requirements like security as well as other design considerations which cannot be included as part of the CloudSim Simulation model. Hence, before buying or provisioning new hardware for this task, an additional cost cutting measure might be to use one of the numerous available virtual test beds [39], [40]. While some of these may come at a cost, the savings from detecting a failure in these test beds are likely to significantly outweigh the cost finding them after implementation and purchasing hardware. Also, in setting up a test bed, a large number of industry proven reference architectures for different kinds of cloud-based systems are available for use [41]. This may provide a useful starting point and save more costs. In the future, we plan to develop strategies for setting up such test beds, both as a way of validating the practical utility of our approach and for providing a rational way of navigating private cloud testing options.

If after setting up the (real or virtual) test beds, the design is found not to match expectations, prior steps in the process might need to be revisited and improved in a new iteration. This is repeated until all stakeholder goals are considered satisficed, perhaps after series of (re)negotiations.

4 DISCUSSION

When compared with other existing techniques [10], [42] for using simulation in early stage system development, the approach described in this paper provides three key benefits. First, it considers a proper treatment of multiple stakeholders' goals as its starting point and

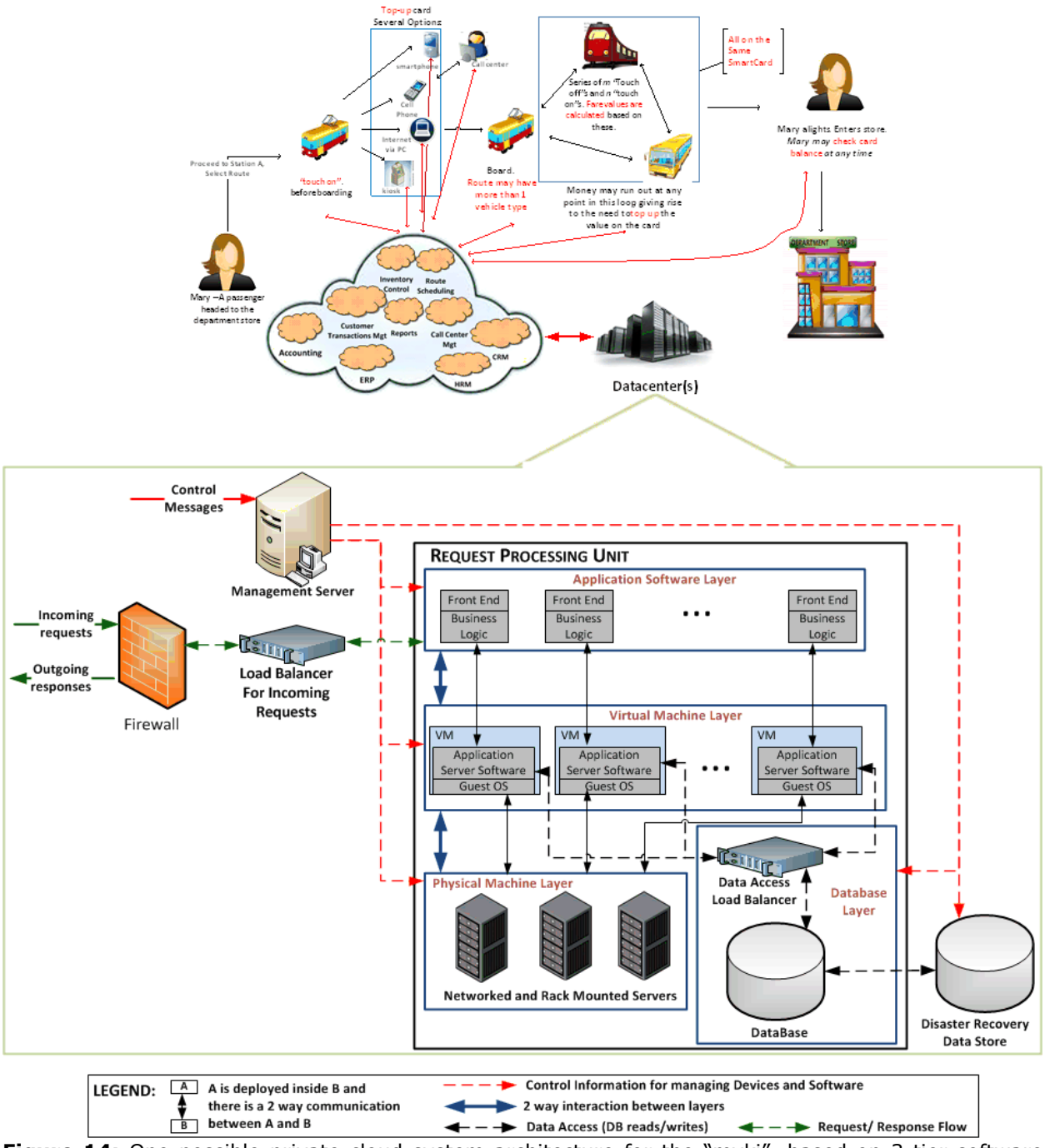


Figure 14: One possible private cloud system architecture for the "myki", based on 3-tier software architecture. Numbers, layout and capacities of devices in the "RPU" are derived directly from simulation for various workloads. Other components added to keep with Private Cloud best practices.

uses this as a frame of reference throughout the entire process, giving rise to higher confidence that the design will be more likely to meet the goals of intended stakeholders. Secondly, it gives more room for human judgment and domain knowledge to guide progress through every phase, providing designers with greater control and flexibility. Third, it relies on analysis techniques that are already well known and used in the Software Engineering community, thus the learning curve required to use the approach is expected to be minimal. Determining the cause and resolution of conflicts in the requirements engineering process for cloud-based systems can be tricky because they could arise solely from the goals, from

the stakeholders themselves, from the system's domain or the interplay of these and other factors. This is even more complicated when the multi-dimensional, multi-stakeholder, multi-objective and large scale nature of cloud-based systems is factored in. This work shows one qualitative way of dealing with this interconnectedness of problem sources towards better resolution of such conflicts while attempting to prevent them from manifesting in the eventually developed system.

As an aid to the requirements negotiation process, we expect that this work would provide a more accurate decision support than verbal negotiation and conflict resolution since simulations are actually used to provide more objective evidence of how design decisions affects the goals of each stakeholder in the cloud-based system development process. Another benefit of using simulation in this approach is that since real-world request (inter)arrival rates and other factors are impossible to determine in reality, the designer can evaluate how the system behaves under different known probability distribution functions.

Without a systematic approach such as the one described in this paper, evaluating whether cloud-based system architecture will be good enough to meet stakeholder goals could be quite challenging, expensive and time consuming. For example, this may involve purchasing as well as manually/physically configuring a cloud and/or testing it. However, from a science of design perspective, two major concerns that still need to be more properly addressed in order to make this approach more robust are:

i. How can convergence after very few steps be ensured?
ii. Do the incrementally better architectural alternatives considered via the simulation model actually approach the global "optima" in the design space?

Addressing these issues will be important in making sure that the modeling costs involved in using the proposed approach does not get so significant as to outweigh the potential time and cost reduction. Although our current results look promising, an essential part of our future work will entail the incorporation of other techniques (e.g., [10]) into our approach which may help ensure quick convergence. Providing tool support for our approach is also another avenue by which modeling costs may be further reduced, and one that we are also considering, going forward.

In Software Engineering, an important part of quantitatively complementing qualitative Goal-Oriented analysis frameworks has been to formally define sound and complete axiomatization that unambiguously define how reasoning might be done about the quantitative goals. Prior work on this include those which discuss quantitative reasoning in the NFR Framework [43], quantitative extension to the NFR Framework to aid requirements traceability [44] and formal reasoning about alternatives in KAOS goal models [45]. In this paper, we gave heuristics for reasoning about levels of goal and stakeholder satisfaction when SIGs are quantitatively extended with design constraints obtained from the domain. Hopefully, this will be a good starting point for the formalizations needed to reason about the quantitative extensions to the SIG as presented in this paper.

Finally, we wish to re-emphasize that, in this paper, we presented the 6 steps in our approach in sequence purely for pedagogic reasons. Such a purely incremental/sequential approach is neither intended nor recommended to be followed in practice. Rather, practitioners may run multiple steps in parallel, revisit earlier steps based on outcomes of latter steps or alter the sequence - whatever is more suited to the nature of each project. The iterative, interleaving and interactive nature of our approach, as intended, was discussed earlier in Section 3.

5 RELATED WORK

A recent goal-oriented simulation technique [10] attempts to fully automate the search through the design space for a design that results in an optimal degree of goal satisfaction.

This technique utilizes a multi-objective optimization algorithm and converges on a set of Pareto-optimal solutions which approach the global optima within the sample space considered. This technique, however, requires significant amount of mathematical/probabilistic modeling and extension of the goal model with numerical values prior to the automation steps. It therefore has the disadvantage that it may not scale very well to very large goal models with a lot of interrelationships since the amount of mathematical modeling required prior to automation may grow exponentially with the size and complexity of the goal model. Also, since the sample of the design space considered is randomly generated and automatically searched, it "robs" human designers of the opportunity to guide the process – something that may be crucial for very critical systems. Our approach allows human architects to guide the process. We have been experimenting with using this Genetic Algorithm based technique [10] complementarily with the one described in this paper, since each one seems to address the other's perceived shortcomings. It is noteworthy that the inequality expressions in the various constraint symbols (Section 3.4.2) bear some semblance to the objective functions used to guide the algorithmic search through the design space [10]. We hope to build on this similarity, among others, in creating a hybrid from these two approaches, if possible.

Another approach for using simulation to evaluate non-functional aspects of cloud-based system designs uses results from queuing theory to predict whether a cloud-based system with a fixed capacity will remain stable [42] over time. Stability is measured by the ability of the system to maintain constant response times as more and more requests arrive in the datacenter. However, evaluations are validated against theoretical queuing models such as the M/M/1 model, as opposed to actual stakeholder goals that necessitated such designs in the first place. Considering the multi-stakeholder, multi-objective nature of cloud-based design, it is not clear how such evaluations can be used for improving the system in a rational and fair manner.

In the non-cloud domain, the Architecture Simulation Model (ASM) Approach [7–9] has been proposed for use in evaluating system design alternatives with respect to specific non-functional goals like performance and scalability in both the design and maintenance phases of system development. The ASM originally proposed the integration of simulation, which is quantitative in nature with goal-orientation, which is qualitative in nature. It therefore served as an important the basis for the work described in this paper.

Designing a system to be fair to multiple stakeholders is a well-studied topic in the Software Engineering community. The importance of stakeholder identification and analysis in the design process [16], [24], [46] as well as techniques for dealing with conflicts that arise from the requirements of different stakeholders [47 – 49] have been described. In addition, the topic of negotiating among stakeholders in order to resolve conflicts have also been discussed [50], [51]. Integrating more of these results into our approach is likely to make it even more useful.

The work in this paper is related to early stage modeling, analysis and evaluation of system design which have also been well studied. Architecture validation during the analysis phase of system development via the Enterprise Architecture Simulation Environment (EASE) [52] and early stage performance modeling in the design of embedded control software [53] are some of the topics that have received attention in this area. Regarding the use of non-functional requirements for early-stage performance evaluation of designs, an ontology-aided performance engineering approach using the NFR Framework has been proposed [54].

Although Capacity Planning is expected to be less important in cloud-based systems than it has been for traditional enterprise systems, it can still be a critical success factor in an organization's Cloud Computing Adoption strategy. To this end, several work on capacity planning for cloud-based systems have appeared [55–57]. However, these have been mostly focused on datacenter resource optimization techniques with little or no

consideration of how decisions to use such techniques can affect the stakeholders of the cloud-based system. This work can help capacity planners to better integrate actual stakeholder needs into their work, considering the conflicts that may exist among such stakeholders and their goals.

6 CONCLUSION

The goal-oriented simulation approach proposed in this paper starts with the capturing and understanding of multiple stakeholders' goals, which are subsequently refined and quantitatively complemented, based on certain domain characteristics. A CloudSim simulation model is then built based on these, as the means of assessing the impact of design choices on the degree to which the various goals are satisficed. After a number of iterations the approach yields a design which may then be tested in a real cloud deployment. Other contributions made by this paper include a new visual notation to help in managing the complexity of the modeling process as well as heuristics for reasoning about the quantitative additions to the SIG, which is a qualitative goal modeling framework.

Our approach demonstrates one way of exploring, evaluating, and selecting among cloud-based system design alternatives with respect to stakeholder goals. This is likely to provide better rational decision support and even better cost savings for Cloud Computing, which seems to be among the most critical technological innovations for cost savings, while resulting in architectures that are more likely to be good enough despite the unique nature of cloud based system design and conflicts arising from stakeholders and their goals. Using this approach, architects can more rationally and systematically transition from stakeholder goals to the architectural design of a cloud computing-based system. The work also demonstrates the value of goal-oriented i.e. a rational approach to the science of design, especially when combined in an interleaving manner with simulation, in understanding and conquering the complexity involved in developing a cloud-based system.

We are currently working on how our approach can be applied to other layers in the XaaS model, in addition to the Infrastructure-as-a-Service (IaaS) layer explored in this paper, including the Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) layers. In the future we plan to investigate how to make the approach converge much faster to satisficing architecture, while developing guidelines for ensuring that the design in each iteration is better than the ones in the previous iterations. Additionally, we plan to conduct a case study in which the results obtained from our goal-oriented simulation approach are compared with values obtained from the actual running system, as a way of further validating our approach. As a matter of fact, we have already started to run experiment, currently in the Google App Engine environment, which should help enhance the level of confidence in the validity of the results of the simulation runs presented in this paper.

7 REFERENCES

[1] M. Armbrust, A. D. Joseph, R. H. Katz, and D. A. Patterson, "Above the Clouds: A Berkeley View of Cloud Computing," UC Berkeley, Tech. Rep., No. UCB/EECS-2009-28, 2009

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, vol. 53, no. 6, p. 50, 2009.

[3] R. Clarke, "User Requirements for Cloud Computing Architecture," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 625–630.

[4] S. Zardari and R. Bahsoon, "Cloud adoption: a goal-oriented requirements engineering approach," in Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, 2011, pp. 29-35.

[5] T. J. Lehman and S. Vajpayee, "We've Looked at Clouds from Both Sides Now," SRII Global Conference, 2011, pp. 342–348.

[6] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in Proceedings. Fifth IEEE International Symposium on Requirements Engineering,

2001, pp. 249-262.
[7] T. Hill, S. Supakkul, and L. Chung, "Confirming and Reconfirming Architectural Decisions on Scalability: A Goal-Driven Simulation Approach," in On the Move to Meaningful Internet Systems: OTM 2009 Workshops, 2009, pp. 327–336.
[8] T. Hill, "Software maintenance and operations hybrid model: An IT services industry architecture simulation model approach," in Research Challenges in Information Science (RCIS), Fifth International Conference on, 2011, pp. 1–6.
[9] T. Hill, S. Supakkul, and L. Chung, "Run-time monitoring of system performance: A goal-oriented and system architecture simulation approach," in Requirements@Run.Time (RE@RunTime), First International Workshop on, 2010, pp. 31-40.
[10] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in 19th IEEE International Requirements Engineering Conference (RE'11), 2011, pp. 79–88.
[11] A. Kertész, G. Kecskeméti, and I. Brandic, "Autonomic SLA-aware Service Virtualization for Distributed Systems," in 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2011, pp. 503–510.
[12] R. Jeyarani, R. V. Ram, and N. Nagaveni, "Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 585-586, 2010.
[13] "Official Home Page of the myki project." [Online]. Available: http://www.myki.com.au/. [Accessed: 31-Oct-2011].
[14] A. C. Yoh, H. Iseki, B. D. Taylor, and D. A. King, "Interoperable transit smart card systems: Are we moving too slowly or too quickly?" Transportation Research Record: Journal of the Transportation Research Board, vol.1986, no.1, pp.69–77, 2006.
[15] N. Mallat, M. Rossi, V. Tuunainen, and A. Öörni, "An empirical investigation of mobile ticketing service adoption in public transportation," Personal and Ubiquitous Computing, vol. 12, no. 1, pp. 57–65, 2008.
[16] H. Sharp, A. Finkelstein, and G. Galal, "Stakeholder identification in the requirements engineering process," in Proc. Tenth International Workshop on Database and Expert Systems Applications, pp. 387–391, 1999.
[17] A. Anton, "Goal-based requirements analysis," in Proceedings of IEEE International Requirements Engineering Conference (RE'96), pp. 136-144, 1996.
[18] "IBM Cloud Computing Reference Architecture," 23-Jul-2010. [Online]. Available: https://www.ibm.com/developerworks/mydeveloperworks/blogs/c2028fdc-41fe-4493-8257-33a59069fa04/tags/ccra?lang=en. [Accessed: 08-Mar-2012].
[19] "NIST Cloud Computing Reference Architecture." [Online]. Available: http://www.nist.gov/manuscript-publication-search.cfm?pub_id=909505. [Accessed: 09-Aug-2012].
[20] "The CHAOS report." [Online]. Available: http://www.projectsmart.co.uk/docs/chaos-report.pdf. [Accessed: 08-Mar-2012].
[21] L. Chung and JCSP Leite, "On non-functional requirements in software engineering," Conceptual modeling: Foundations and Applications, vol. 5, no. 4, pp. 285-294, 2009.
[22] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach," Software Engineering, IEEE Transactions on, vol. 18, no. 6, pp. 483–497, 1992.
[23] E. S. K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in Proceedings of the Third IEEE International Symposium on Requirements Engineering, pp. 226–235, 1997.
[24] A. van Lamsweerde, "Requirements Engineering: From System Goals to UML Models to Software Specifications", John Wiley & Sons, Chichester (2009)

[25] E. Yu, "Modelling strategic relationships for process reengineering," in Social Modeling for Requirements Engineering, 2011, pp. 1-15.
[26] R. Buyya, R. Ranjan, and R.N. Calheiros, "Modelling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities", International Conf. on High Performance Computing & Simulation, pp.1-11, 2009.
[27] A. Nuñez, J. Vázquez-Poletti, A. Caminero, J. Carretero, and I. Llorente, "Design of a New Cloud Computing Simulation Platform," Computational Science and Its Applications-ICCSA 2011, pp. 582–593, 2011.
[28] D. Kliazovich and P. Bouvry, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," in IEEE Global Telecommunications Conference (GLOBECOM 2010), pp. 1-5, 2010.
[29] "Point of Sales Systems (POS) - GAO Research." [Online]. Available: http://www.gaoresearch.com/POS/pos.php. [Accessed: 08-Mar-2012].
[30] J. D. C. Little and S. C. Graves, "Little's Law," Operations Management, pp. 81-100, 2008.
[31] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, "Profiling and modeling resource usage of virtualized applications," in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 366–387, 2008.
[32] "TPC-C - Homepage." [Online]. Available: http://www.tpc.org/tpcc/. [Accessed: 08-Mar-2012].
[33] R. N. Calheiros, R. Ranjan, and C. D. Rose, "Cloudsim: a novel framework for modeling and simulation of cloud computing infrastructures and services," Arxiv preprint arXiv:, pp. 1-9, 2009.
[34] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446-452, 2010.
[35] A. Greenberg, J. Hamilton, and D. Maltz, "The cost of a cloud: research problems in data center networks," ACM SIGCOMM Computer, vol. 39, no. 1, pp. 68-73, 2008.
[36] R. Kazman, M. Klein, and M. Barbacci, "The architecture tradeoff analysis method," Proc. 4th IEEE International Conference on Engineering of Complex Computer Systems, pp. 68 - 78, 1998.
[37] L. Chung, B. A. Nixon, and E. Yu, "Using non-functional requirements to systematically select among alternatives in architectural design," in Proc. 1st Int. Workshop on Architectures for Software Systems, pp. 31–43, 1995.
[38] "Mapping Applications to the Cloud." [Online]. Available: http://msdn.microsoft.com/en-us/library/dd430340.aspx. [Accessed: 08-Mar-2012].
[39] A. I. Avetisyan et al., "Open cirrus: A global cloud computing testbed," Computer, vol. 43, no. 4, IEEE, pp. 35–43, 2010.
[40] "Cloud computing research testbed | Technical overview." [Online]. Available: http://www.hpl.hp.com/open_innovation/cloud_collaboration/cloud_technical_overview.html. [Accessed: 10-Mar-2012].
[41] "Intel® Cloud Builders Secure and Efficient Cloud Infrastructure." [Online]. Available: http://www.intel.com/content/www/us/en/cloud-computing/cloud-builders-provide-proven-advice.html. [Accessed: 10-Mar-2012].
[42] J. Wang and M. N. Huhns, "Using simulations to assess the stability and capacity of cloud computing systems," in Proceedings of the 48th Annual Southeast Regional Conference, 2010, p. 74.
[43] P. Giorgini, J. Mylopoulos, and E. Nicchiarelli, "Reasoning with goal models," in Proceedings of the 21st Int. Conference of Conceptual Modeling (ER2002), no. October, pp. 167 – 182, 2003.

[44] J. Cleland-huang, W. Marrero, and Brian Berenbach, "Goal-Centric Traceability?: Using Virtual Plumblines to Maintain Critical Systemic Qualities," IEEE Transactions on Software Engineering, vol. 34, no. 5, pp. 685-699, 2008.
[45] A. van Lamsweerde, "Reasoning about Alternative Requirements Options," Conceptual modeling: Foundations and Applications, LNCS, vol. 5600, pp. 380 - 397, 2009.
[46] L. Chung, D. Gross, and E. Yu, "Architectural design to meet stakeholder requirements," In: Donohue, P(ed.). Software Architecture. Kluwer Academic Publishers. San Antonio, Texas.
[47] A. V. Lamsweerde and R. Darimont, "Managing conflicts in goal-driven requirements engineering," Software Engineering, vol. 24, no. 11, pp. 908-926, 1998.
[48] B. Boehm and H. In, "Conflict analysis and negotiation aids for cost-quality requirements", Univ. of Southern California, Tech. Rep., No USC-CSE-99-530, 1999.
[49] A. Finkelstein, M. Harman, and S. Mansouri, "'Fairness Analysis in Requirements Assignments," Requirements, pp. 115-124, 2008.
[50] H. In, D. Olson, and T. Rodgers, "A requirements negotiation model based on multi-criteria analysis," in Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, pp. 312–313, 2001.
[51] H. In, B. Boehm, T. Rodgers, and M. Deutsch, "Applying WinWin to quality requirements: a case study," in Proceedings of the 23rd International Ionference on Software Engineering, 2001, pp. 555–564.
[52] S. S. Brink, "Enabling Architecture Validation in the Analysis Phase of Developing Enterprise or Complex Systems using Enterprise Architecture Simulation Environment (EASE)," in Military Communications Conference, 2007, pp. 1–8.
[53] S. Wang and K. G. Shin, "Early-stage performance modeling and its application for integrated embedded control software design," Proceedings of the fourth International Workshop on Software and Performance (WOSP '04), p. 110, 2004.
[54] P. P. Sancho, C. Juiz, R. Puigjaner, L. Chung, and N. Subramanian, "An approach to ontology-aided performance engineering through NFR framework," Proceedings of the 6th International Workshop on Software and Performance, pp. 125 - 128, 2007.
[55] S. Mylavarapu, V. Sukthankar, and P. Banerje, "An optimized capacity planning approach for virtual infrastructure exhibiting stochastic workload," in Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), 2010, pp. 386 - 390.
[56] R. Lopes, F. Brasileiro, and P. D. Maciel, "Business-Driven Capacity Planning of a Cloud-Based IT Infrastructure for the Execution Of Web Applications," IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1-10.
[57] D. A. Menasce and P. Ngo, "Understanding cloud computing: Experimentation and capacity planning," in Computer Measurement Group Conference, 2009.
[58] L. Chung, B. Nixon, E. Yu and J. Mylopoulos, "Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers", Dordrecht (2000)
[59] L. Duboc, E. Leiter, D. Rosenblum, "Systematic Elaboration of Scalability Requirements through Goal-Obstacle Analysis", *Software Engineering, IEEE Transactions on*, vol.PP, no.99, pp.1, 2011
[60] C. B. Weinstock and J. B. Goodenough, "On systems scalability," Software Engineering Institute, Technical Note CMU/SEI-2006-TN-012, 2006.
[61] Menasce, D., Vigilio, A., "Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning", Prentice Hall PTR, Englewood Cliffs (2000)

A GOAL-ORIENTED SIMULATION APPROACH FOR OBTAINING GOOD PRIVATE CLOUD-BASED SYSTEM ARCHITECTURES

Lawrence Chung[1], Tom Hill[2], Owolabi Legunsen[1], Zhenzhou Sun[1], Adip Dsouza[1], Sam Supakkul[3]

[1,2]The University of Texas at Dallas, 800 West Campbell Road, Richardson, TX 75080, USA [1]{chung, ool090020, zxs101020, amd061000}@utdallas.edu, [2]tom.hill.fellow@gmail.com [3]ssupakkul@ieee.org

~~[1,2]The University of Texas at Dallas, 800 West Campbell Road, Richardson, TX 75080, USA [1]{chung, zxs101020, amd061000, ool090020}@utdallas.edu, [2]tom.hill.fellow@gmail.com~~

~~[3]Keane Inc., 5600 Tennyson Parkway, Suite 265 Plano, TX 75024, USA. ssupakkul@ieee.org~~

ABSTRACT

The fast-growing Cloud Computing paradigm makes it possible to use unprecedented amounts of computing resources at lower costs, among other benefits such as fast provisioning and reliability. In designing a good architecture – the numbers, types and layouts of devices – for a cloud-based system, which meets the goals of all stakeholders, such goals need to be factored in from the earliest stages. However, there seems to be a lack of methodologies for incorporating stakeholder goals into the design process for such systems, and for assuring with higher confidence that the designs are likely to be good enough for the stated goals. In this paper, we propose a goal-oriented simulation approach for cloud-based system design whereby stakeholder goals are captured, together with such domain characteristics as workflows, and used in creating a simulation model as a proxy for the cloud-based system architecture. Simulations are then run, in an interleaving manner, against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives. We illustrate important aspects of this approach for the private cloud deployment model and report on our experiments, using a smartcard-based public transportation system.

Keywords: Cloud Computing, System Architecture, Goal-Oriented, NFR Framework, Simulation Model, CloudSim, Little's Law, Requirements Engineering

1. INTRODUCTION

Would it be possible to predict whether a cloud-based service will indeed be good enough from the perspectives of multiple stakeholder goals, such as cost, performance and scalability? Can we do this in the early stages of development, before committing to the potentially costly and time-consuming implementation and testing? Answering these questions affirmatively is important to people who want to take advantage of the many benefits, such as cost reduction, fast service provision, reliability, and the like, promised by Cloud Computing [1], ~~[2],~~ [~~3~~2]. In particular, designers of proposed cloud-based systems will want to investigate how to come up with designs ~~and whether their designs will~~to meet the goals of all stakeholders before it is too late, disruptive or expensive to make ~~any~~ desired changes. This paper proposes one approach for doing this by using goal-orientation together with~~,~~ cloud computing simulations that are cheap and quick to set up, since goal-oriented techniques allow for capturing stakeholder goals and using them in exploring, analyzing and selecting among architectural design alternatives. In the proposed approach, goal orientation and simulation are used in an interleaving, iterative manner ~~such that~~ - goal-oriented aspects can be carried out simultaneously, and in any order with, simulation modeling and analysis~~different aspects of goal-orientation can be carried out simultaneously, and in any order with, simulation modeling and analysis~~.

By nature, early stage design in cloud-based systems is multi-stakeholder, multi-objective, multi-dimensional and large scale. It is "multi-stakeholder" in the sense that there are different types of stakeholders (e.g., cloud vendors, service providers, and end users) and "multi-objective" in that the goals of one group of stakeholders differ from those of another group and goals are oftentimes competing and conflicting even within the same group. The "multi-dimensional" nature concerns the fact that stakeholder goals need to be simultaneously investigated for different stakeholder groups at different levels of abstraction (hardware level, Virtual Machine (VM) level, database level, etc.). It is also "large scale" because very large numbers of individuals in each stakeholder group need to be considered while designing such systems, necessitating the use of different kinds of workloads in assessing the quality of system design. These characteristics make cloud-based system design quite challenging. Without a systematic method, such as the one proposed in this paper, it would be a daunting challenge to understand, develop, and successfully operate cloud based systems. This goal-oriented, simulation-based approach provides a fast way with little financial and manpower resources to tackle the challenge.

Users' requirements for cloud computing architecture, as well as the role of goal-oriented requirements engineering in Cloud Computing adoption have been described [4][3], [54]. However, there still needs to be a more systematic approach for integrating these into the architecture design process for cloud-based systems, towards attaining the level of rigor and success that has been achieved through the use of goal orientationby goal-oriented techniques in traditional (not cloud-based) Software Engineering. Perhaps, the lack of such a rational approach in cloud-based system design is the main culprit in what has been described as "solutions that are looking for a problem" [65]? Our main aim in this paper is to describe a systematic approach that which may be used to design cloud-based systems based on stakeholder needs.; We borrowing heavily from best-of-class Goal-oriented Oriented Software Engineering [76] techniques, while showing how such might be used in a realistic system design scenario. We also focus on the so-called private cloud deployment model, in which one organization (the Cloud Service Creator) owns and operates the datacenter but gives shared access to other organizations who subscribe on a pay-as-you-go basis.

The use of simulations for investigating complex system behavior is not new. However, in the Software Engineering community, there has been a recent increase in the recognition of the role that simulations can play in the early stages of system design, especially for evaluating and validating design decisions. This is perhaps due to the recognition among researchers that rising levels of systems complexity need to be matched by more reliable ways to more reliably of investigateing designs in the early stagesearly in the process. The marriage of goal-orientation (techniques for exploration, selection among, and evaluation of architectural alternatives based on stakeholders' goals) and simulation (which is fast and easy to set up) is expected to be highly beneficial but challenging. Some work has recently emerged in this regard, e.g., [8] for using simulation to confirming and reconfirming architectural decisions using simulations [7], [9], [10] for runtime monitoring and system maintenance [8], [9], and [11] for simulating and optimizing design decisions in quantitative goal models [10]. The Cloud Computing field has also been quick to recognize the role of simulations in investigating infrastructural level concerns [12–14][11], [12]. Our use of simulation is for investigating the impact of Cloud-Computing infrastructural design choices on the proposed system's ability to meet the needs of its stakeholders.This work seeks to use simulation for investigating cloud-based system design concerns that are at a higher level of abstraction than what has been done, and closer to the needs of a system's stakeholders.

At a high level of description, our approach starts with the capture stakeholder goals plus some domain characteristics such as workflows. The goals are then refined and extended

quantitatively with numbers obtained from the domain, using estimation methods that we propose. All these are subsequently used to create a simulation model as a proxy for the cloud-based system architecture. Lastly, simulations are run iteratively, in an interleaving manner, against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives that have a better chance of meeting the stakeholder goals.

The main contributions of this paper are as follows:

1. ~~This paper includes the following contributions:~~ A method of combining goal-oriented requirements engineering techniques with simulation for cloud computing,
2. A technique for complementing the qualitative goal models in the NFR Framework [58] with a quantitative approach
3. The development of an interactive, iterative and interleaving simulation technique based on those quantitative extensions as well as stakeholder needs
4. A new visual notation, along with heuristics and guidelines for reasoning quantitatively about the goal models and their quantitative extensions.

In the rest of this paper, Section 2 provides a background to the approach, including an overview of the real world system to be used for illustration and some key Goal-Oriented System Engineering concepts to be used in the rest of the paper. Section 3 gives step-by-step details of our proposed approach as applied to the system under study. We discuss our experimental outcomes in Section 4 and related work in Section 5. We conclude and give thoughts on future directions in Section 6.

~~The rest of the paper is as follows. In Section 2, we give a background to the approach, including an overview of the real world system to be used for illustration and some key Goal-oriented System Engineering concepts to be used in the rest of the paper. Section 3 gives step-by-step details of our proposed approach as applied to the system under study.~~

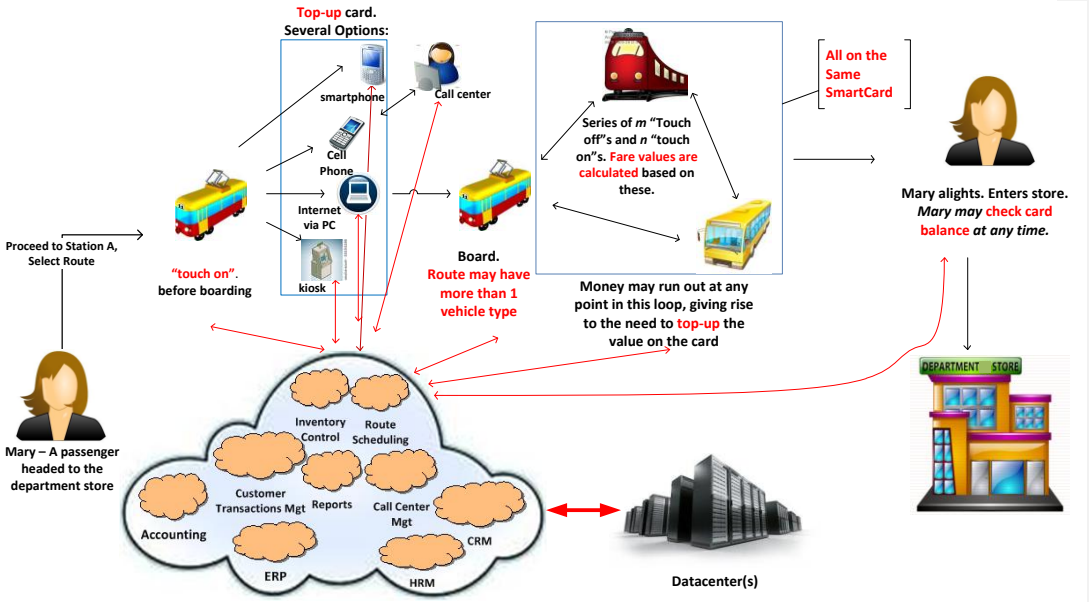


Figure 1: In the "myki", very many transactions must be processed in real time. ***How do we come up with a cloud-based design to satisfy this and design constraints for all stakeholders?***

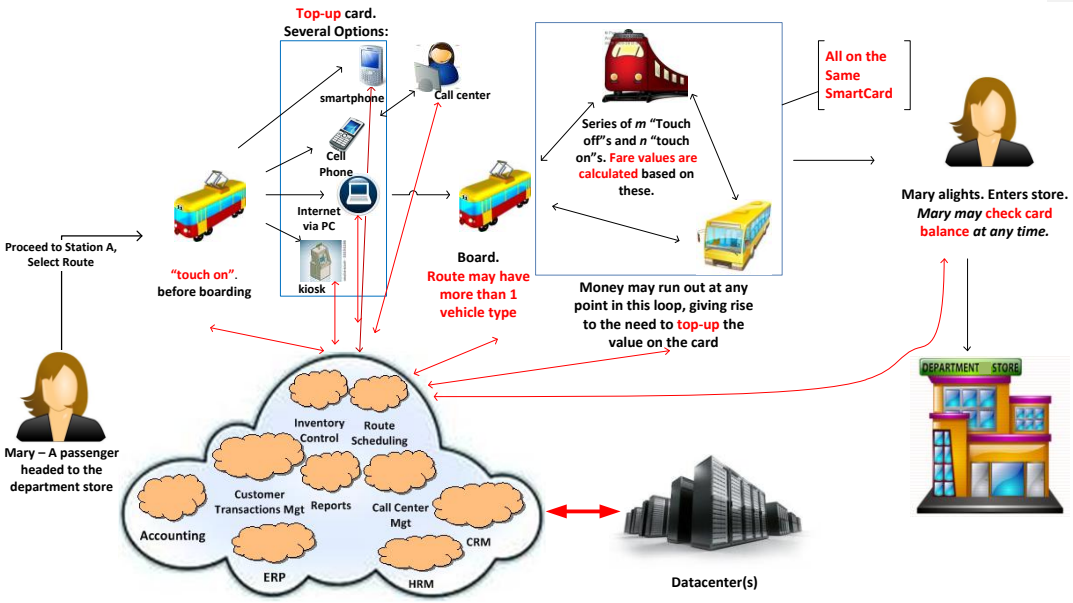


Figure 1: In the "myki", very many transactions must be processed in real time. ***How do we come up with a cloud-based design to satisfy this and design constraints for all stakeholders?***

~~We discuss our experimental outcomes in Section 4 and related work in Section 5. We conclude with thoughts on future directions in Section 6~~.

2. BACKGROUND

2.1 Application Domain Overview: The "myki"

The "myki" [~~15~~13] is a contactless smartcard system, recently created to automate and harmonize ticketing on all channels of public transport (trains, buses and trams) in Victoria, the most densely populated State in Australia. 547 million passengers boarding were reported on all modes of transportation in 2010 and the tram system in Melbourne is the largest in the world. Fig. 1 shows some essential end-user related activities within the "myki". These include adding value to smartcard (Fig. 1, Top Up), swiping card to enter vehicle (Fig. 1, Touch On), swiping card to leave vehicle (Fig. 1, Touch Off) and the automatic computation and deduction of fares. Other transactions include checking the card balance, viewing past transactions, reporting and blocking a lost card, getting periodic reports and system initiated third party payment transactions to banks. All transactions require real time communication with, and processing in, the data center. ~~Thus the~~The ability of the cloud datacenter to handle such large workloads is an important question that all stakeholders will like answers to, beforehand, if possible.

The authors think that as Victoria seeks to remain one of the most livable places in the world, future improvements to the myki system are likely to become more and more important to all stakeholders. An examination of the issues associated with such a system [~~16~~14] as well as projections of possible future improvements for such a system ~~(e.g.,~~

[~~17~~15]~~)~~ might cast the myki project as a possible candidate for migration to the cloud, in our opinion. To further motivate the use of the "myki" system as an example of how cloud-based system architecture might be designed, we consider the hypothetical cases in which (1) Melbourne earns the right to host the Summer Olympics (2) The Federal ~~government~~ Government of Australia votes to adopt the "myki" for all public transportation in all of Australia. Technical considerations for such futuristic assumptions will likely lead architects to consider whether proposed designs will scale to the resulting ~~foreseen and unforeseen~~ increases in workload, beyond what was originally ~~planned for~~deployed. Questions about cost and performance will also become more critical as design considerations. These goals interact and can be in conflict. For example, as end users demand higher performance levels at lower prices, the costs to service providers may increase. In this paper, we focus on such scalability, performance and cost goals and show how our approach might be used to derive an architecture that is more likely to be good enough in the presence of multiple stakeholder groups, whose goals are also usually incompatible with one another.

2.2 Our Perspective on Cloud Computing

~~Because "Cloud Computing" is still evolving, there are many definitions of what the term might mean[2], [3], [18–21], with little or no consensus among them. While it is not our aim in this paper to provide yet another definition, it is important to state our perspective on cloud computing, as this drives our notion of what the design of systems entails. It is pertinent to state that this paper considers the private cloud deployment model exclusively. The following NIST definition of a private cloud[3] is perhaps the most widely known and cited:~~

~~"A private cloud ... infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise."~~

~~This definition bears semblance to the well known collocation and outsourcing service models. It is therefore not a surprise that the cloud computing paradigm shares similar motivations with these.~~ Assuming a virtualized datacenter, our view of what constitutes a cloud is as follows:

> "A (private) cloud consists of a collection of virtual machines running on hosted infrastructure (including servers, processors, storage and network) owned by one stakeholder group who gives shared access to many customers in other stakeholder groups"

In Section 2.3, we elaborate more on the important stakeholder groups in Cloud Computing but the diagram in ~~Figure~~ Fig. 2 shows a side-by-side depiction of the most essential concepts in the "myki" system with those in the cloud, based on our stated perspective on cloud computing.

2.3 Stakeholders in Cloud-Based System Development

In coming up with a fair system ~~which is fair~~ with respect to meeting the goals of intended users, identifying the stakeholders is critical to the development process [~~22~~16], [~~23~~17]. In their respective Cloud Computing Reference Architecture (CCRA) proposals, both IBM and NIST have proposed the different stakeholder groups that should be identifiable in any cloud-based system development project [~~24~~18], [~~25~~19]. We follow the IBM perspective because it seems to be based on their experience in actually designing and building cloud-based solutions for their many clients. Also, if adopted by The Open Group (to which IBM has submitted a proposal)~~it has been proposed~~, we feel that the IBM CCRA has a higher chance of becoming an industry standard. The IBM CCRA identifies the following groups of stakeholder roles:

i. Cloud Service Consumer

This is an organization, a human being or an IT system that consumes service instances delivered by a particular cloud service. The service consumer may be billed for its interactions with cloud service and the provisioned service instance(s).

ii. Cloud Service ~~Creator~~Provider

The Cloud Service Provider has the responsibility of providing cloud services to Cloud Service Consumers. A cloud service provider is defined by the ownership of a common cloud management platform (CCMP). This ownership can either be realized by truly running a CCMP by himself or consuming one as a service.

iii. Cloud Service ~~Provider~~Creator

The Cloud Service Creator is responsible for creating a cloud service, which can be run by a Cloud Service Provider or directly exposed~~and by that~~ exposed to Cloud Service Consumers. We note that the Cloud Service Provider may comprise two distinct roles: a Service Development Architect responsible for modeling and deploying the application in the datacenter and a Service Delivery Architect responsible for running the application, mapping Virtual Machines (VMs) to hardware infrastructure and providing a cost model.

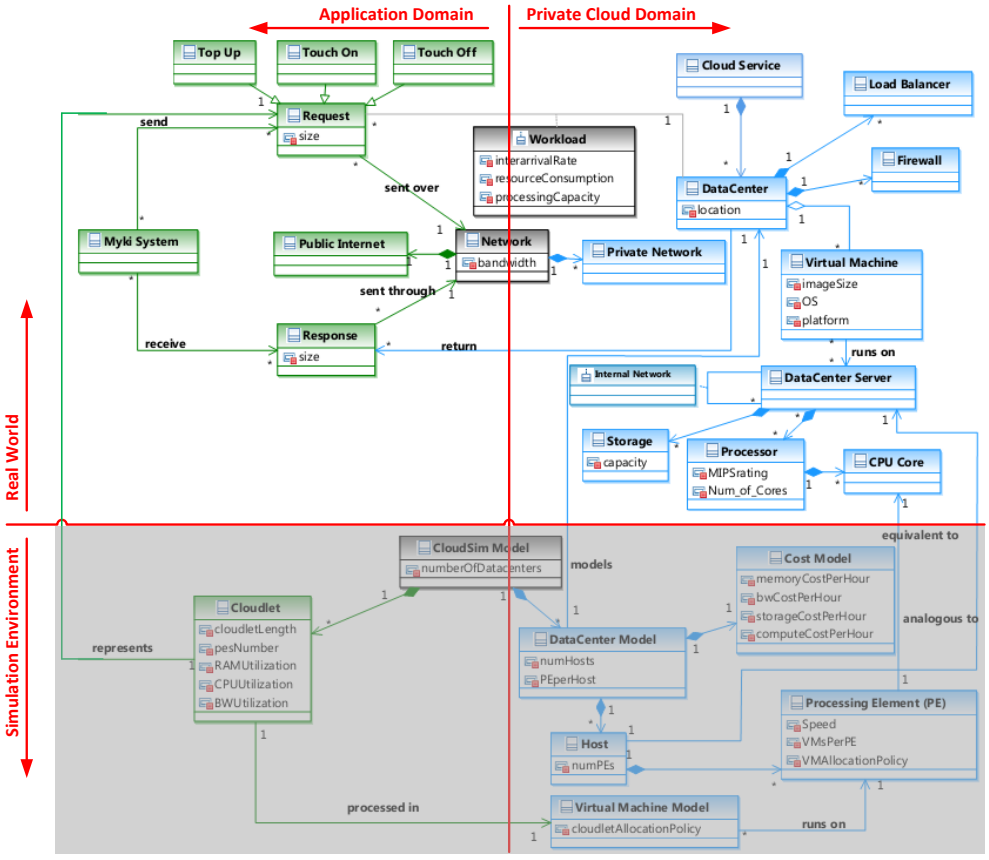


Figure 2: High-level view of the most essential concepts in the "myki" (left) and our view of a private cloud (right) for the real-world (top) and Simulation environments (bottom)

iv. End User

It is interesting to note that the End User is not treated as a separate stakeholder group in neither the NIST nor IBM CCRA. The reason for this omission is not stated but it seems that both CCRAs assume that the requirements of the End User are known to the Cloud Service Consumer who then uses these as a basis of selecting the services in the first place. We depart from this thinking and treat the End User as a separate stakeholder group, following the well-established fact from Software Engineering that the number one source of project failures is a lack of understanding of the End User's requirements [2620].

~~Figure~~ Fig. 3 shows the hierarchy of these groups of stakeholders while Table 1 describes them within the "myki".

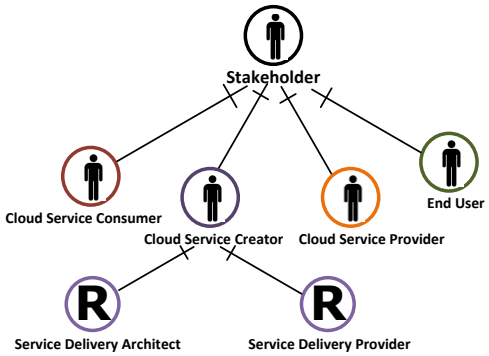


Figure 3: A Hierarchy of Stakeholders and roles in Cloud-Based System Design

2.4 Goal-oriented System Engineering with NFR Framework

A closer look at the "Goals" column in Table 1 reveals that most of the goals are expressed in subjective, non-functional~~[27]~~ terms [21]. This is typical and the problems associated with the elicitation, analysis and specification of stakeholder requirements expressed in terms of such non-functional goals has been studied and described in the Software Engineering literature (~~(e.g.,~~ [~~7~~6], [~~28–30~~22] and [~~36~~23]). Also, Frameworks like KAOS [~~31~~24], i* Framework [~~32~~25] and the NFR Framework [~~28~~58] have been proposed for dealing with stakeholder goals. The Softgoal Interdependency Graph (SIG) proposed as part of the

Role	Entity	Description	Goals
End User	Passengers	Patrons who need to pay fares to travel from one location to another and carry out essential activities in their daily lives	Quick and convenient request processing.
Cloud Consumer	KAMCO	They consume cloud services in order to provide services for end users to carry out their activities within the system.	Costs should not affect profit margins.
Cloud Provider	CSP	A fictional Cloud Service Provider, CSP Inc., is assumed in this paper.	Good server utilization factor. Right brokering and resource allocation policies.
Cloud Creator	HP	Owns and runs the Datacenter which is backbone of the private cloud.	Meeting SLAs agreed to with Cloud Service Providers.

Table 1: Identifying Cloud Based System Development Stakeholders in the "myki" system

NFR Framework is useful for depicting, analyzing and reasoning about softgoals - goals that are addressed not absolutely but in a good enough sense. These kinds of goals are the focus of this work (although we focus on performance, scalability and cost for clarity reasons and because of space constraints). Hence, SIGs are used subsequently. A high level SIG for some stakeholder goals in the myki system is shown in Fig. 4. Stakeholders are shown as agents and roles in the spirit of the i* Framework.

In SIGs, softgoals are shown as clouds with labels - a type and topic pair - beneath each cloud. Roughly, the type of a softgoal describes it while its topic (shown in square brackets) tells what it applies to. Softgoals may be refined by AND/OR decompositions. OR decompositions show design alternatives, where achieving one or more alternatives satisfices the parent softgoal. All alternatives in an AND decomposition must be achieved to satisfice the parent softgoal. The bottom leaves in a SIG should ultimately be *operationalizing softgoals*, represented as thick clouds and which can be actualized by an assigned agent in the domain. SIGs assist in reasoning about the interactions and conflicts among the stakeholders' goals by depicting the degree of positive or negative contributions among softgoals with pluses (+) and minuses (-). For example, ~~which shows, that~~ the softgoal, P*rofitability[cloud service]* may be~~is~~ "helped" by the softgoals, *Cost Effective[myki system]* and *Maximize[datacenter utilization].*

SIGs also allow for showing the level of importance of a goal relative to others in the goal model so that a goal marked with "!!" is considered very critical, one marked with "!" is critical and one that is not marked with any explanation is neutral.

As an example, Fig. 4 also shows the conflict between the (very critical) softgoal, *!!Fast Response Time[Datacenter]* and *High Utilization[DataCenter Servers]* which is needed to improve profitability. This is so because the goal, *!!Fast Response Time[Datacenter]* is very critical to end-users and, hence, the Cloud Consumer will require that the system should have very good response times whenever requests are sent through the "myki" system. However, the goal, *High Utilization[DataCenter Servers]*, which is important to the profitability of the Cloud Service Creator, entails making such architectural decisions like using less powerful servers or adding more load to existing servers instead of buying new ones. Such decisions will save cost but are likely to have a negative impact on the *!!Fast Response Time[Datacenter]* softgoal. Hence Fig. 4 shows that the goal, *High Utilization[DataCenter Servers]* (indirectly) helps the goal *Profitability[Cloud Service]* but hurts the goal *!!Fast Response Time[Datacenter]*. Similar reasoning is behind the positive and negative contributions among the other goals shown in Fig. 4. The technical challenge is how to come up with a private cloud architecture which is good enough inspite of such incompatibilities.

~~As an example, Fig 4 also shows the conflict between the (very critical) softgoal, *!!Fast[response time]* and *Profitability[cloud service]* i.e. profitability. The goal, *!!Fast[response time]* is very critical to end-users and, hence, the Cloud Consumer will require that the system should have very good response times whenever requests are sent through the "myki" system. However, the goal, *Maximize[datacenter utilization]*, which is important to the profitability of the Cloud Service Creator, entails making such architectural decisions like using less powerful servers or adding more load to existing servers instead of buying new ones. Such decisions will save cost but are likely to have a negative impact on the *!!Fast[response time]* softgoal. Hence Fig. 4 shows that the goal, *Maximize[Datacenter Utilization]* helps the goal *Profitability[cloud service]* but hurts the goal *!!Fast[response time]*. Similar reasoning is behind the positive and negative contributions among the other goals shown in Fig 4. The technical challenge is how to come up with a private cloud architecture which is good enough in the presence of such incompatibilities.~~

2.5 Using CloudSim for Cloud Based System Design

CloudSim [~~33~~26] has been proposed and used as a tool for simulating and investigating cloud computing environments, based on the following premise: [~~12~~], [~~34~~], [~~35~~]. ~~According to the authors of CloudSim [33],~~

"The core hardware infrastructure services related to the Clouds are modeled in the simulator by a Datacenter component for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on Datacenter's host components. By VM processing, we mean a set of

operations related to VM life cycle: provisioning of a host to a VM, VM creation, VM destruction, and VM migration"

We use ~~it~~ Cloudsim to investigate concerns at a higher level of abstraction, closer to the goals of stakeholders. CloudSim was chosen mainly because, to the best of our knowledge, there was no other cloud simulation tool available when the research began. Other cloud simulation tools ~~(i.e., [36], [37])~~ have ~~recently~~ since been proposed [27], [28]

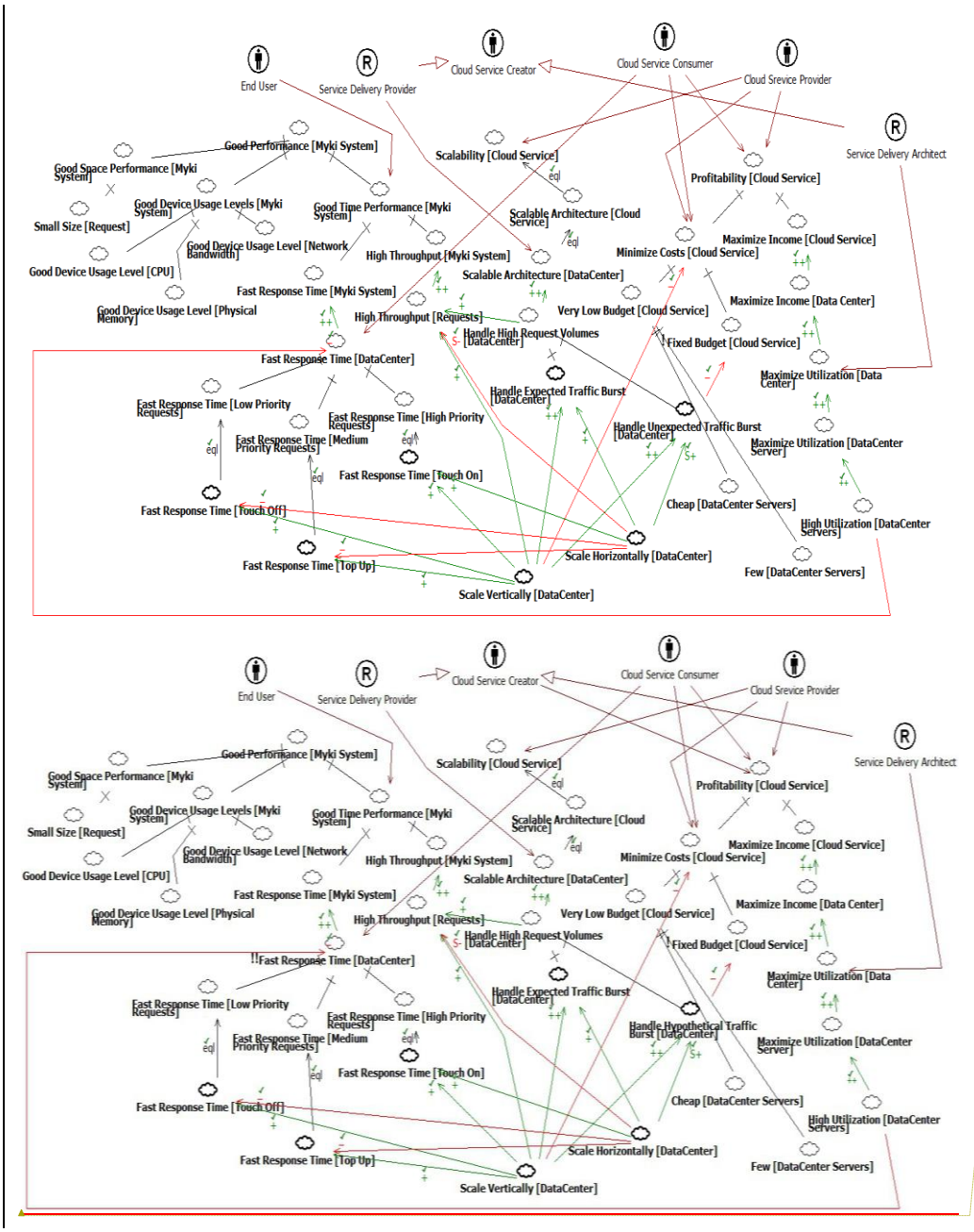


Figure 4: Softgoal Interdependency Graph (SIG) showing some high-level "myki" stakeholder goals

Formatted: Font: (Default) Verdana, 10 pt

and will be investigated in the future. The two relevant CloudSim concepts with which we are concerned in this paper are:

i. Cloudlets

In CloudSim, Cloudlets may be used to represent an application or a request to an application. Cloudlets consume assigned resources and utilize them for a time calculated as a length in Million Instructions (MI) divided by the VM performance in Million Instructions per Second (MIPS)[1]. This paper uses cloudlets as single requests that arrive for processing in the datacenter.

ii. Datacenter

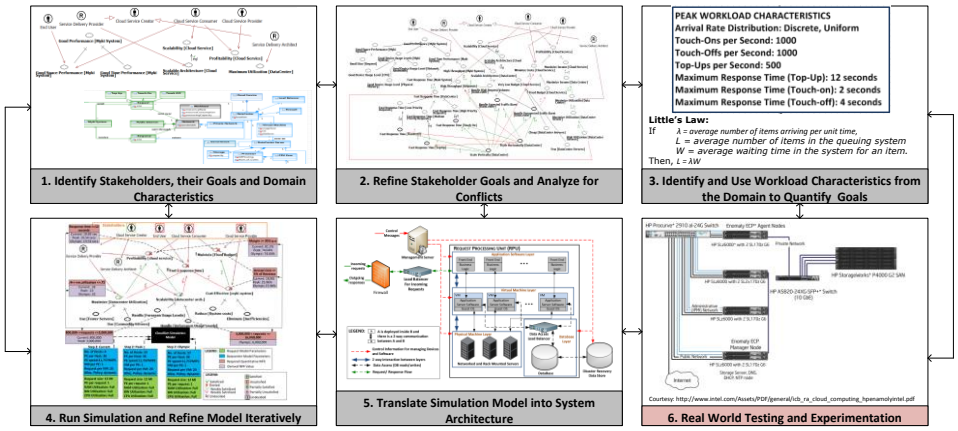
In CloudSim, a data center is modeled as an aggregation of computing hosts which, in turn, consist of Processing Elements (PE). A PE is analogous to a CPU core. VMs are deployed on PEs in the simulator.

In a way, a Cloudlet object in CloudSim encapsulates the left-hand side of the domain model in ~~Figure~~ Fig. 2 while a Datacenter object can be made to represent the right hand side. This is so because it is the generation of requests by users in the application domain that lead to design constraints on the architecture of the system to be deployed in the cloud – essentially a virtualized datacenter according to our stated view of what constitutes (private) clouds. Hence using CloudSim for ~~system~~ cloud-based system design ~~will~~ involves three main phases:

a. Obtaining relevant information like stakeholder goals, usage patterns and constraints from the application domain;
b. Mapping the information from (a.) into a simulation model in CloudSim which is then used for investigating whether various configurations and workloads can meet stakeholder goals; and
c. Mapping results from (b.) back into a system design deployment in the cloud

Section 3 addresses how all these phases are dealt with for the "myki~~", as a realistic system design scenario~~.

3. THE PROPOSED APPROACH



~~Figure 5:~~ ~~A summary of our 6-step process.~~

[1] Descriptions from CloudSim Authors obtained from comments at http://groups.google.com/group/cloudsim

3.1 Overview

This approach is a 6-step process which starts with an identification of stakeholders and their goals and ends with a design that is considered more likely to be good enough in the presence of conflicting stakeholder goals. Additionally, application domain characteristics are obtained and translated into constraints on the system design using the CloudSim simulation model. Simulations are used to confirm and reconfirm the quality architectural decisions at different stages of the process.

~~Figure~~ Fig. 5 shows a summary of the 6 steps, which are not meant to be followed in strict sequence. Rather, they are envisioned for use in an interactive, interleaving and iterative manner whereby aspects of multiple steps may be carried out in parallel while revising outcomes of earlier steps where necessary, depending on new information gained from the later steps. Also, the steps represent a single experimental run that may be carried out iteratively until a good enough design is obtained. The results presented later in this paper do not cover Step 6, which is part of our future work.

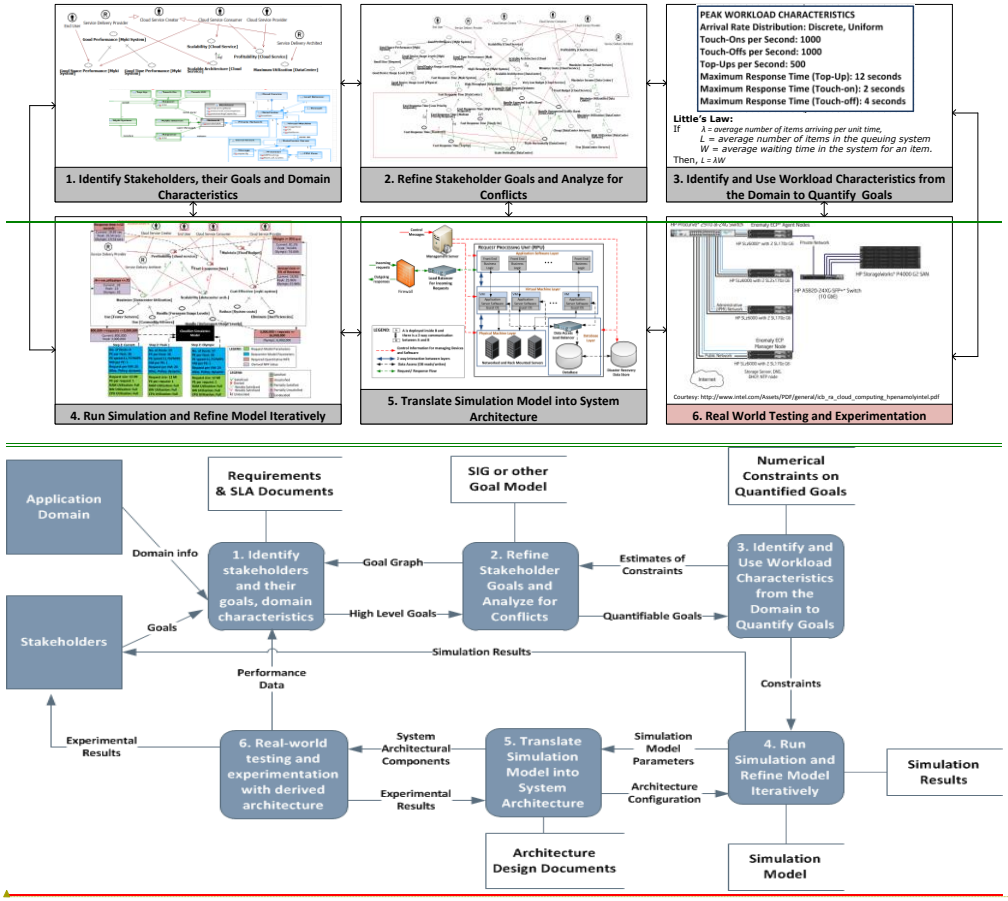


Figure 5: A summary of our 6-step process. All steps are described in the rest of this section.

Formatted: Font: Verdana, 10 pt

3.2 Step 1: Stakeholder and Goal Identification

Proper identification of stakeholders and their goals is the basis of a successful design process - one which results in a system that has a better chance of meeting the goals of all its intended users. Typically, the goals of stakeholders will be discovered, analyzed and refined during the requirements elicitation phase of the project. Techniques for such goal-oriented elicitation for cloud-based system development have been proposed~~, e.g., [4]~~[3], [~~5~~4]. Table 1 summarizes the results of such elicitation for the 4 stakeholder groups discussed in Section 2.3, for the "myki".

3.3 Step 2: Analyzing Stakeholder Goals for Conflict Identification

Having identified the stakeholder goals, the next step is to represent the goals, along with the stakeholders interested in them in the form of a SIG diagram. Other frameworks can also be used but we have selected the SIG for reasons specified in Section 2.4. Fig. 4 shows one such representation wherein stakeholder(s) interested in a particular goal, as well as any specific roles that they play in the system, are shown alongside the high-level goal model. In Fig. 4, stakeholders are represented as agents (circled stick symbol) and roles (circled R) in the spirit of the i* Framework. The resulting diagram is henceforth referred to as an Agent-SIG. Subsequently, the Agent-SIGs are simpler than Fig. 4for clarity in presenting the interaction between stakeholders, their goals, simulation results and the interrelationships between these.~~We have kept the Agent-SIG at a sufficiently high level in order to more clearly see the interaction between stakeholders, their goals and the interrelationships between these.~~ In practice, ~~subsequent~~ even more model refinement would be required, whereby each high level goal is sufficiently AND/OR decomposed until specific operationalization, assignable to an agent in the domain is derived.

3.4 Step 3: ~~Using Information from the Domain to Quantify High Level Goals~~Quantitatively Augmenting the Qualitative Goal Model with Domain Specific Information

In order to use simulation, which is quantitative, to evaluate ~~qualitative~~ the impact of design decisions on the degree of satisficing of softgoals, it is necessary to augment the ~~the analysis of~~qualitative goal model (Agent-SIG) from ~~Section 3.3~~Step 2, shown in Fig. 4, with numbers that guide decision making.~~. To do this,~~This entails the translation of stakeholders' requirements from the application domain ~~have to be translated~~ into numerical target values that the eventual system design is expected to meet or exceed for the related softgoals to be considered satisficed. We refer to these target values as "design constraints~~. that the system must satisfy.~~ Typically, these constraints will be evident, at least partially, during the requirements elicitation phase for a cloud-based development project [~~5~~4] and can be of different types. In this paper, ~~W~~we focus on design constraints related to cost, scalability and performance goals ~~constraints and~~ . This section shows how ~~constraints on~~ initial ~~-~~ estimates of design constraints on these goals ~~goals~~ may be obtained, ~~using~~ in the "myki" example. ~~These numerical~~ These estimates ~~ions of design constraints~~ will be used ~~are used initially~~ to ~~formulate~~ create a baseline design which is then improved upon in subsequent~~ly~~ iterations.

One question concerns the impact of errors in the initial estimates on the number of iterations needed for our interactive technique to converge. We plan to experimentally investigate this in the future. However, conventional wisdom, based on experience with iterative techniques in Numerical Analysis, seems to suggest that the number of iterations in the overall process will tend to increase in direct proportion to the errors in the initial estimates. To this end, we feel that special care must therefore be taken while estimating the design constraints. Also, to start the process of estimation, one softgoal needs to be selected. We suggest the selection of a softgoal that is deemed very critical to the end-user ("customer is king") and/or a softgoal which allows the creation of a queuing or similar performance model.

It is important to point out that the aim of this step is not to replace any of the reasoning or elaboration activities and techniques normally associated with goal models. Neither is it meant to be a proposal for a new kind of quantitative goal model. Rather, it complements existing techniques in order to further facilitate the use of goal-orientation with simulation. The utility of the SIG produced in Step 2 to the activities in Step 3 includes:

1. The SIG can help stakeholders negotiate and agree on target values that define what it means for the goals in the SIG to be considered good enough.
2. It helps requirements engineering personnel to discoverc what domain specific phenomena need to be quantified and fed into the simulation model.
3. It serves as the basis of reasoning about the degree to which the softgoals have been satisficed.

In the rest of this section, we show how the performance, scalability and cost goals for the "myki" were quantitatively augmented, based on the refinements shown in the SIG of Fig. 4.

~~Special care must therefore be taken while estimating because the number of iterations of the overall process will tend to increase in direct proportion to the error in creating this starting point. Also, to start the process of estimation, one softgoal must be selected. We suggest the selection of a softgoal that is deemed very critical to the end user ("customer is king") and/or a softgoal which allows the creation of a queuing or similar performance model.~~

3.4.1 Quantifying Performance Goals

In refining the high-level performance softgoal, *Good Performance[Myki System]*, as shown in the Agent-SIG of Fig. 4, we adopted the classic NFR Framework definitions and approach to dealing with performance goals, referred to as the *Performance sort* [22], [54]. To start the estimation process, ~~For the high level SIG in Fig. 4,~~ we considered the end-user goal, *!!Fast[response times]* ~~is considered~~to be ~~the most~~ very critical from the End User's perspective. ~~and is, t~~Thus, this softgoal is selected initially. We note, however, that this choice is by no means unique and will vary from project to project. ~~selected to start with. Next, we refine this high level goal till we arrive at satisficing operationalizing softgoals.~~ Next, we try to answer the question of what it might mean for the performance related softgoals in the qualitative goal model ~~For this goal to~~ to be considered satisficed.

For the "myki" one key performance objective is for~~,~~ the system ~~must~~ to process ~~all~~ individual transactions within the smallest amount of time possible while still processing the maximal volume of requests per ~~unit time~~day. This objective fits well into the interaction among *space performance*, *time performance* and *device utilization* as espoused in the *Performance sort* technique and captured in the Agent-SIG of Fig. 4. But, how do we derive these quantities like number of simultaneous transactions, maximum processing time per request, and total number of requests per day which, among others, form the key *performance variables* that are of interest to stakeholders? The maximum allowable processing time for each request will ~~usually~~ ideally be stated in the SLA ~~agreed upon before project commencement or during requirements elicitation~~agreements. For the sake of discussion in this paper, we take this value to be *12 seconds*, comparable to what is common in most POS systems [~~38~~29]. For the number of requests per day, we found that the current workload on the "myki" system is 800,000 requests per day (rpd). This value, combined with other domain knowledge from the existing system is then used along with Little's Law, as shown in Section 3.4.1.1~~below~~, to compute how many simultaneous requests the datacenter must be able to handle ~~simultaneously~~ in order ~~so as~~ to meet the

performance goal of *12 seconds* average response time.

Realistically, a more refined treatment might be desirable in identifying and deriving numerical values for the related performance variables. For instance, instead of computing the processing times or number of requests per day for all requests as an aggregate like we do in illustrating our ideas in this paper, these values may be computed for the individual types of request (i.e. touch on, touch off, top-up) ~~-~~individually. All the estimation techniques discussed in the rest of this section will remain applicable, and the increase in the amount of computations are expected to be offset by the benefits to performance engineers in terms of a more detailed view of the system under investigation.

3.4.1.1 Estimating Performance Variables using Little's Law

Given the way that we are choosing to measure the performance goal for the "myki", as described in the previous section, we need a way to relate the different performance variables together mathematically. In our case, we found Little's Law [~~39~~30] to be useful. Little's Law defines a relationship among a queuing system's average arrival rate, average processing time and average throughput. Little's Law states that,

If

λ = average number of items arriving per unit time,

L = average number of items in the queuing system and

W = average waiting time in the system for an item.

Then,

$$L = \lambda W \qquad (1)$$

Because cloud-based systems will consist of requests that are eventually processed in a datacenter, with some reasonable assumptions, we think that it should be possible to model most (private) cloud-based applications as queuing systems over which Little's Law can be applied to gain intuition for creating the simulation model to be used during system design.

3.4.1.2 Some Assumptions made in using Little's Law for the "myki'

In using Little's Law to create a queuing model for the arrival of requests in the datacenter, the following assumptions were made about the system:

i. The datacenter is a single channel server.
ii. The datacenter will handle all requests uniformly.
iii. Data needed to process requests is completely and accurately contained within the datacenter.
iv. The response time for a request is equal to the processing time of each request in the datacenter, neglecting latencies in client devices and the network.
v. Multiple datacenters may conceptually be treated as a single datacenter, assuming the right task scheduling and resource allocation policies among them.

These assumptions make it possible to model the entire "myki" as a queuing system in line with the fundamental assumptions made by Little's Law. In addition to these assumptions, the datacenter is initially treated as a black box into which requests flow and out of which responses come after a certain amount of processing time. The internals of the datacenter are then considered as the model is improved subsequently.

3.4.1.3 Calculations and Estimates

In computing the smallest amount of time possible while still processing the maximal volume of requests per unit time, the following facts from the domain, adapted from information obtained from actual "myki" engineers, were used:

- The system currently handles 800,000 requests per day (rpd)

- 75% of requests arrive during, and are split equally between, the morning and evening rush hour periods. The remaining 25% are evenly spread over the rest of a 12-hour day.
- Rush hour is between 0700-0930 and 1700-1900 on weekdays in Melbourne.

Current average request arrival rates per day are as follows:

- *Morning rate = 300,000 requests ÷ 2.5hours = 120,000 requests per hour*
- *Evening rate = 300,000 requests ÷ 2hours = 150,000 requests per hour*
- *Non-rush hour rate = 26,667 requests per hour*

To compute the average number of requests to be simultaneously processed in the datacenter under any of these workloads, we assume that request inter-arrival rates have a discrete uniform distribution that is constant per minute and let

L = average no. of requests processed simultaneously by the datacenter;
W = average processing time per request = 1 minute = 0.01667 hour;
λ = arrival rate of requests to the datacenter = 150,000 requests/hour.

By *(1)*,

$$L = \lambda W = 150{,}000 \text{ requests/hour} \times 0.01667 \text{ hour} = 2500 \text{ requests}$$

This implies that, to handle the same workload as the current system, the cloud-based system should, on the average, be able to handle 2500 requests simultaneously. For other design scenarios, such as the hypothetical nationwide adoption of the "myki" system, similar calculations can be used to find out how many requests the system must be able to process simultaneously. This value is then used subsequently to form input to the simulation model. Instead of randomly selecting and testing designs from the design space, the designer is, thus, handed a more precise starting point. Note that we have assumed a uniform distribution for the request inter-arrival rates for simplicity and brevity and other probability distribution functions should be investigated in practice.

3.4.2 Quantifying Scalability Goals

The concept of a scalability goal has recently been more precisely defined in terms of specified scaling assumptions [59], wherein a *scalability goal* is

> *"a goal whose definition and required levels of goal satisfaction make explicit reference to one or more scaling assumptions".*

In turn, a *scaling assumption* is

> *"a domain assumption specifying how certain characteristics in the application domain are expected to vary over time and across deployment environments".*

We found this definition and related treatment of scalability more suitable for our goal-oriented simulation approach, since it allows us to evaluate possible system behavior based on domain assumptions. There are other definitions of scalability as a system goal (e.g., [60]) which may be more germane to other kinds of projects.

In order to use this definition, however, the question arises as to what domain characteristic(s) the scaling assumption(s) should be based on. We restrict our focus in this paper to workload-related domain characteristics. Specifically, at the risk of appearing to use a rather crude measure of scalability, we will only consider how the system behaves under varying workloads (measured in terms of requests per day, rpd). Our rationale is that the aim in this paper is not to show how to properly treat individual softgoals, but to show the impact of design decisions on the degree to which multiple softgoals are simultaneously satisficed. More detailed treatment of scalability as a softgoal may be found in the Software Engineering literature [59], [60]. Also, in considering a variation-in-workload approach to scalability, we sought to see the ability of the system under investigation to handle spikes in the amount of load that it supports.

We define our *scaling assumptions* in terms of four kinds of workloads outlined in Table 2. ~~The current system is known to process 800,000 requests per day (rpd) but is expected to scale to 3,000,000 rpd at peak capacity at maximum capacity.~~ The *Current* workload refers to the amount of load currently handled by the "myki" while the *Peak* workload represents the expected load at peak capacity. The values used for the *Current* and *Peak* are somewhat obfuscated from real data obtained from "myki" operators. ~~Also~~Next, in order to show how the cloud might be used to scale this system even further, the *Olympic* workload~~we~~ considers the case where the Cloud Service Consumer ~~(KAMCO in the "myki") also wants~~ needs the system to ~~be able to~~ handle temporarily large spikes in usage as may occur, if ~~for example if~~ Melbourne were to host the Summer Olympics in the near future. Lastly, the *Australia* workload seeks to see how scalable the system might be for ~~Lastly, we investigate how to design such a system for~~ a very large scale permanent expansion in its use, as may occur ~~in the purely hypothetical case that~~if every state in Australia decides to adopt the "myki" ~~instead of building smartcard ticketing systems from scratch~~. Both the *Olympic* and *Australia* scenarios are purely hypothetical although it should not be hard to see how they may correspond to real-world use cases. All four workload types represent the *scaling assumptions* that we have made and are by no means exhaustive. ~~For brevity, we shall refer to the workloads in these four cases by the names shown in the first column of Table 2, which also shows the calculated number of requests the system will be expected to handle in each case as well as how those numbers were derived.~~ \

One factor that needs to be accounted for is whether other system goals are expected to have fixed or changing target values as the workload varies. Cases where system goals are expected to remain fixed have been defined as *scalability goals with fixed objectives* (SGFO) while those in which the system goals can vary have been defined as *scalability goals with varying objectives* (SGVO) [59][2]. For the "myki", one SGFO is that ~~We hypothetically consider the architecture to be scalable if it can~~ the system should be able to handle ~~(un)foreseen spikes~~all scaling assumptions ~~in request volume~~ while maintaining VM utilization ~~at~~ between 40% and 75%, where utilization is computed as the fraction of time during which each VM is busy. An example of SGVO would be that some reasonable amount of performance degradation in terms of longer response times might be expected for the *Australia* scaling assumption.

Workload	Requests/day	Justification
Current	800,000	Statistics from current system
Peak	3,000,000	Expected workload at capacity
Olympics	6,000,000	Assuming that influx of tourists causes use of public transport to double
Australia	16,814,521	If Melbourne current usage is representative of national patterns, this number is proportional to the peak capacity that the current system is expected to handle.

Table 2: Different workloads to which the "myki" is expected to scale in the cloud

3.4.3 Quantifying Cost Goals

Cost constraints can be viewed from the perspective of multiple stakeholder groups. For the "myki", passengers care about how much they are charged per transaction as well as how much taxpayer money the Government is spending to support the system. Cloud Consumers want to make enough money from fares to offset costs incurred in using the cloud service. Cloud Service Providers and Cloud Service Creators will similarly want to

[2] The abbreviations are not part of the original definitions, but have rather been used for brevity in the rest of the paper.

ensure that the cost of the services provided are offset by what they charge to their respective clients. We assume that annual cash inflow from fare collection for the *Current* workload is about $100,000,000, that revenues in other workloads are directly proportional and that fare collection is the only revenue source. Will cloud usage make the system ~~more profitable~~ cost-effective/profitable to all stakeholders? Will the system be able to sustain itself without Government help? This is important as it is not likely that fare prices charged to passengers will change merely because the system is migrated to the cloud platform. Also, prices charged by Cloud Service Creators (HP in the "myki" example) are typically fixed. Hence, the system design will need to ensure minimal costs. For simplicity, we consider the system cost effective and profitable for all stakeholders as long as Cloud Computing and Server costs do not exceed 5% of total revenue in the *Current, Peak* and *Olympic* Workloads and 50% in the *Australia* workload.

3.5 Step 4: Interleaving and Iterative Simulation and Model Refinement

The aim in this step is to translate the constraints from Step 3 (Section 3.4) into simulation models that serve as a proxy for the actual cloud-based system, through which the quality of design decisions can be investigated for impact on the satisficing of softgoals within the goal model (SIG). We use an interleaving and iterative method, whereby

3.5.1 Cloud Computing Simulation Modeling with CloudSim

~~According to the authors of CloudSim [33],~~

> ~~*"The core hardware infrastructure services related to the Clouds are modeled in the simulator by a Datacenter component for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on Datacenter's host components. By VM processing, we mean a set of operations related to VM life cycle: provisioning of a host to a VM, VM creation, VM destruction, and VM migration"*~~

~~Thus, i~~In CloudSim, modeling the cloud is essentially an activity in deciding the parameters of a suitable virtualized datacenter, which is then internally converted to a discrete event simulation model ~~(using SimJava [40]).~~ For our experiments, the constraints on the system design, derived from the previous step are used to inform initial parameter selection which is then improved subsequently, based on simulation results and the degree of stakeholder goal satisfaction. Next, we briefly describe initial parameter selection, modeling and simulation model refinement steps.

~~**3.5.1.1 Simulation Model Parameter Selection: Incremental vs Interleaving Methods**~~

~~Some of the CloudSim simulation model parameters to be decided on include the nature and size of requests that arrive in the datacenter, the number and capacities of servers and CPU, the number and size of VMs, among others. One key design consideration is how to select these parameters, given that the design space from which they must be selected is huge, if not infinite. It is tempting to use a purely automated search which starts at a minimum and incrementally combines the parameters until an optimum is found. But, assuming that CloudSim can be extended to perform such a search and that such automated search will converge, this purely incremental approach is not likely to be sufficient, since the decision about what is "good enough" is subjective and usually requires some measure of human judgment in each iteration. Our approach is, therefore designed to be interleaving, whereby multiple steps can be (re)visited, depending on the how well progress is being made towards a good-enough design.~~

3.5.1.1 Mapping from Quantified Goals to CloudSim Model

In Section 2.~~6~~5, we gave a high level introduction to the two main CloudSim concepts that are needed to model a cloud-based system design: Cloudlets and Datacenters. In this

Section, we show how the analysis done so far can be mapped to into a Simulation Model in CloudSim, in terms of these two entities.

3.5.1.1.12.1 Modeling Application Requests as Cloudlets

Table 3 describes some attributes of the Cloudlet class that were used in the simulations. The *cloudletLength* attribute represents the number of CPU instruction cycles consumed by a request. Existing techniques for estimating resource usage in virtualized environments (e.g., [41]) do not use such fine grained parameter and work mainly for existing systems. Our approach focuses on early stages of development. Hence, the challenge is to reliably ~~approximating~~ approximate this low-level machine and ~~software~~ implementation dependent variable without writing or deploying any code. We propose some techniques for approximating this value, depending only on information that can be ~~more accurately~~ estimated at design time:

i. If the system exists and it is possible to obtain/estimate the number of instructions currently being processed per second, the server hardware being used and the utilization of the server, then the number of instructions in a request may be approximated from:

$$\frac{\text{MIPS rating of the server}}{\text{Requests/sec at 100\% utilization}}$$

Attribute	Value	Meaning/Rationale
cloudletLength	12 MI	Obtained from calculations
pesNumber	1	1 CPU core of the right capacity is sufficient to process this cloudlet.
cloudletFileSize	1913 bytes	Typical file size at checkout for most e-commerce sites
utilizationModelCPU	Full	Each cloudlet will use an entire CPU core
utilizationModelRam	Full	Same meaning as in CPU
UtilizationModelBw	Full	Same meaning as in CPU

Table 3: Some Cloudlet Properties for a Single Request. Meanings of variables available in the CloudSim API online

ii. If the server on which the system is being run can be determined but there is no way to obtain/estimate the requests per second, then the tpmC metric of the server may be obtained from the TPC-C benchmark[~~42~~32] and used in the formula given in (i). In this case, however, the complexity of the transactions in the system under study has to be estimated as a fraction or multiple of those in the TPC-C (or similar) benchmark.

iii. If the requests per seconds can be obtained/estimated but not the MIPS rating of the actual server used, then the MIPS rating of a server with the closest semblance to the one being used may be used as an approximation

iv. If neither the MIPS rating nor the number of requests can be estimated, the problem may be narrowed down by taking a look at the architecture application and estimate the number of instructions in each component.

For the myki system, assuming HP ProLiant BL680c G7 servers running on Intel Core i7-980X processors (147,600 MIPS) process the 800,000 requests per day at the common server utilization of 15%, calculations yield 11.07 million instructions (MI) per request. Using the peak value 3,000,000 requests per day under the same assumptions yielded a value of 2.95 MI/request. We take this as a range within which the actual value may lie. Calculations using the method in (ii) above yielded a value 7.01 MI/request. Interestingly, this is within the range obtained from the first method. This gives some initial confidence about the estimation methods and suggests that multiple methods may be used complementarily. We use only the worst case computed value of 12 MI per request in the rest of this paper.

3.5.1.~~2~~1.2 Modeling the Data Center Architecture

The practice in the myki system of routing all requests through a single datacenter makes it plausible to model the initial cloud data center as being in single location. The data center configuration shown in Table 4 is the proposed initial design and derives mostly from the estimated design constraints discussed previously. 1-to-1 mapping of VM to CPU core and request to VM have been pessimistically used in coming up with the baseline and will be optimized subsequently.

Property	Value	Explanation
Number of Datacenters	1	
PEs per host	36	Assuming HP ProLiant BL680c G5 Servers running on the Intel Xeon X7460 processor (36 cores).
Number of Hosts	70	70 X 36 = 2520 > 2500, Assuming 1 VM per PE, 1 request per VM.
MIPS per PE	11,757 MIPS	Comparable to the MIPS of a single core in the Intel Core i7 920 (Quadcore) running at 2.66 GHz.
Storage Cost	$0.0134/GB/hr	Extrapolated from Microsoft Azure pricing.
Compute Cost	$0.04/hour	Extrapolated from Microsoft Azure pricing.
Bandwidth Cost	$0.00028/GB/hr EGRESS	Extrapolated from Microsoft Azure pricing.
Memory Costs	$0.05/MB/hr	

Table 4: Initial Datacenter Configuration. (**Host:** a server installation, **PE:** Analogous to a CPU core.)

3.5.2 Using Simulation to evaluate degree of Goal Satisfaction

The multidimensional nature of cloud-based system design requires that the impact of design decisions on goal satisfaction be simultaneously evaluated with respect to the stakeholders, their goals, the proposed system configuration and various workloads that the system is required to support. To better manage this iterative evaluation, we introduce the visual notation in Fig. 6 to help the designer assess impact of particular designs the stakeholder goals. Design constraints (Sections 3.4.2 to 3.4.4) inform the mapping from application domain constraints to simulation model parameters. Hence, the notation in Fig. 6 is proposed to help in visualize the impact of design decisions within the simulation model on the satisfaction of design constraints, in various iteration steps. When combined with the Agent-SIG diagram from Fig. 4, this notation can facilitate reasoning about the various dimensions involved in cloud based-design. We illustrate the use of this notation subsequently, for the "myki" example.

The visual notation works as follows:

i. Simulation model parameters are shown at the bottom of the diagram for different experimental runs, showing both request parameters (in green) and datacenter parameters (in blue), stacked on each other to save space. For example, the leftmost configuration (Run 1) at the bottom of Fig. 5 follows from Tables 3 and 4.

ii. Progression from the simulation model configuration in one experimental run to the next depends on design decisions based on the degree to which all stakeholders' goals have been met. We illustrate this in the "myki" design in subsequent sections.

iii. Design constraints derived from the domain are linked to the goals they impact on by dashed lines. Exact constraints that must be met in order to meet a goal (and satisfy interested stakeholders) are shown in red boxes while constraint values derived from simulation are shown in lilac boxes. Again, related boxes are stacked to save space.

~~iv. Normal rules for reasoning about SIGs remain but the quantities derived from quantative requirements and simulation results are used as a sort of claim softgoal [77] to more properly justify why the softgoals are assigned various degrees of satisfaction.~~

~~v.~~iv. Constraints may be linked to parent goals or to sub-goals. Reasoning about constraints linked to parent goals is as described in (~~i~~vi). When constraints are linked to sub-goals, the impact on the parent goal should be aggregated according to normal SIG reasoning rules.

v. The degree of constraint satisfaction mirrors the degree of goal satisficing in the NFR Framework according to ~~Table 5~~Table 6, depending on how much the simulation results vary from the required constraint.

vi. Normal rules for reasoning about SIGs remain but the quantities derived from quantitative requirements and simulation results are used as a sort of claim softgoal [58] to more properly justify why the softgoals are assigned various degrees of satisfaction.

vii. Reasoning about the constraints and simulation results may be carried out as shown in Table 5. For example, in the Agent-SIG shown in Fig. 8, the response time constraint shows that values obtained from all 3 scenarios are (much) less than the target value of 12 seconds. Hence, the constraint is considered satisfied and (by the mapping of Table 6) the associated softgoal, *!!Fast Response Time[Touch On]*, is considered satisficed. On the other hand, the utilization values obtained from simulation do not lie in the range specified by the constraint which is, thus, unsatisfied. To this end, the softgoal, *High Utilization[Datacenter Servers]* to which this constraint is linked is considered to be *denied*.~~.~~

RULE	FORM		Case 1	Case 2	Case 3	Case 4	Case 5
1	var ≤ y	Result	res < y	res = y	res > y	res >> y	!def(res)
		Constraint	sat	p.sat	p.unsat	unsatisfied	unknown
2	var ≥ y	Result	res > y	res = y	res < y	res << y	!def(res)
		Constraint	sat	p.sat	p.unsat	unsatisfied	unknown
3	x≤var≤y	Result	x<res<y	res = y or res = x	res < x or res > y	res << y or res >> y	!def(res)
		Constraint	sat	p.sat	p.unsat	unsatisfied	unknown
4	var<y, var>y or x<var<y	Treat as the Rule 1,2 or 3 most closely associated, but leave out Case 2					
5	var = y	Result	res = y	res ≠ y	N/A	N/A	N/A
		Constraint	sat	unsatisfied	N/A	N/A	N/A

(Abbreviations: **var**="variable whose value is being measured", **x & y** = "*the (upper or lower) bound on the value of* **var**", **res** = "the actual value of var obtained from simulation results", **sat** = "satisfied", **p. sat** = "partially satisfied", **p.unsat** = "partially unsatisfied")

Table 5: Reasoning about the impact of simulation results on constraint satisfaction

One issue concerns the implicit assumption we have made in the describing the visual notation, viz.: constraint symbols can only link to one softgoal. What if multiple constraints link to the same goal, having different levels of satisfaction? One option might be to extend normal SIG reasoning techniques to handle this situation. This is, however still open to further research.

Softgoal	Satisficed	Weakly Satisficed	Undecided	Denied	Weakly Denied	Unknown
Constraint	Satisfied	Partially Satisfied	Partially Satisfied	Unsatisfied	Partially Unsatisfied	Unknown

~~**Table 5**~~**Table 6:** Levels of Constraint satisfaction versus goals satisficing in the NFR Framework

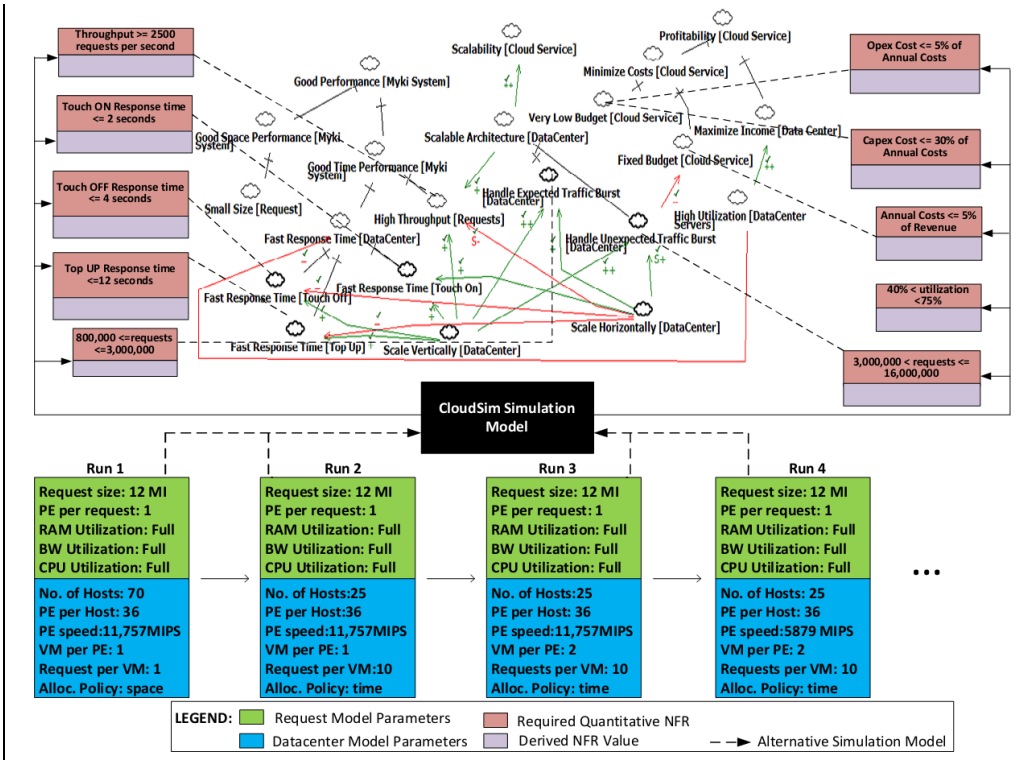


Figure 6: Visual notation for assessing impact of design decisions, represented in the simulation model, on the degree of goals satisfaction during some sample iteration steps.

3.5.3 Initial Simulation, Results and Analysis

For the *Current, Peak* and *Olympic* workloads, using the estimated arrival rates and initial configuration discussed earlier, the first simulation experiment shows a very small, constant processing time and cost per request as shown in Table ~~5~~ 7 and Fig. 7. For the !!*Fast[response times]* softgoal alone, this is a good thing. But, considering other goals, this ~~cannot~~ is unlikely to lead to a fair system for all stakeholders. Annual Cloud Computing costs grow too quickly (cost as percentage of revenue is almost constant) as a 1-to-1 VM to request mapping is used (see iteration 1, ~~Table 6~~Table 7 and Requests per minute and VM Usage graph, Fig. 7). The updated Agent-SIG in Fig. 8 indeed shows that multiple goals in the system have indeed been sacrificed in order to satisfice the critical performance softgoal while Agent-SIG in Fig. 9 shows which stakeholders' goals is (un)met. Steps to improve this initial design are discussed next.

I	Workload	Hosts	VMs	RPM	Policy	p.time	p.cost	a.p.cost p	a.s.cost	t.a.cost
1	Current	70	2500	2500	space	0.14	2158.61	4,043,766.76	1,400,000.00	5,443,766.76
	Peak	261	9375	9375	space	0.14	8306.14	15,560,056.15	5,220,000.00	20,780,056.15
	Olympics	521	18750	18750	space	0.14	16612.3	31,120,112.31	10,420,000.00	41,540,112.31
2	Current	4	125	2500	DW	19.85	141.01	18,693,518.32	80,000.00	18,773,518.32
	Peak	14	469	9375	DW	19.54	738.18	97,096,234.02	280,000.00	97,376,234.02
	Olympics	27	938	18750	DW	19.54	1476.37	194,193,783.39	540,000.00	194,733,7839
3	Current	7	250	2500	time	0.6	216.84	1,701,061.25	140,000.00	1,841,061.25
	Peak	27	938	9375	space	0.1	1021.63	1,307,129.33	540,000.00	1,847,129.33
	Olympics	53	1875	18750	time	0.6	2048.6	16,070,808.36	1,060,000.00	17,130,808.36

(Abbreviations: **I**="Iteration", **RPM** = "Requests per minute", **Policy** = "Cloudlet Allocation Policy", **DW** = "Dynamic Workload Allocation Policy", **p.time** = "Processing time per request", **p.cost** = "processing cost for all RPM", **a.p.cost** = "Annual Processing Cost", **a.s.cost** = "Annual Server Cost", **t.a.cost** = "Total Annual Cost")

~~**Table 6**~~**Table 7:** Simulation results for 3 iterations over the *Current, Peak* and *Olympics* workloads for different configurations. Annual costs are extrapolated from simulation times.

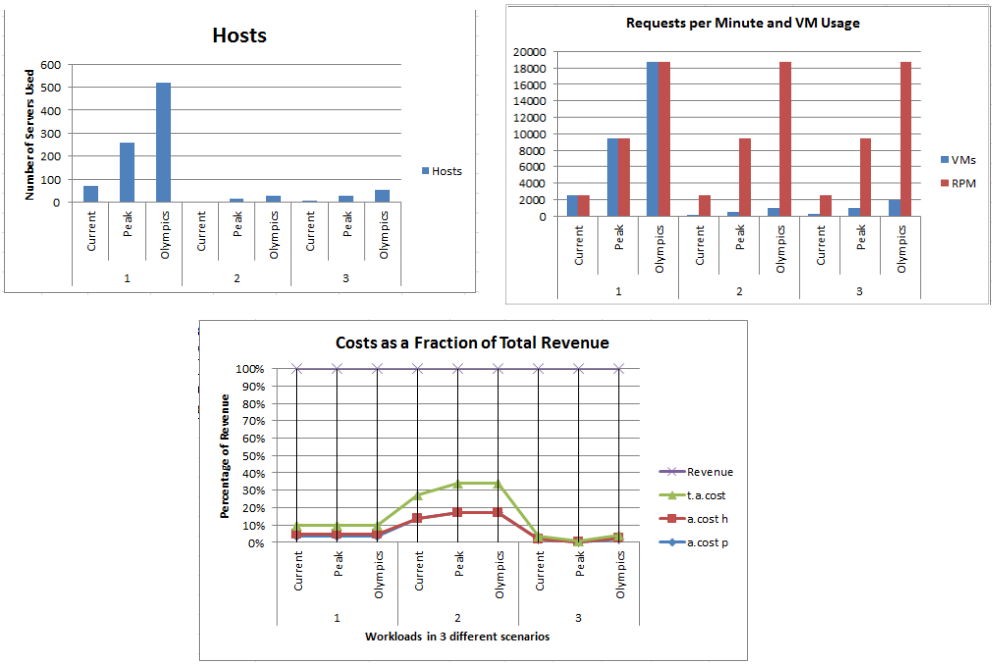


Figure 7: Graphical view of some of the results in ~~Table 5~~Table 6

3.5.4 Model Refinement

Based on the simulation results in the first run, the current configuration, if deployed, is not likely to result in a fair system from the viewpoint of all stakeholders. Hence, the ~~simulation model~~ system design, as captured in the simulation model, will need to be improved towards meeting the goals of as many more stakeholders as possible. To do this, new architectural decisions are made based on insights from the initial results as well as the currently unmet stakeholder goals. For instance, the following decisions were made to improve the initial design for the *Current* workload:

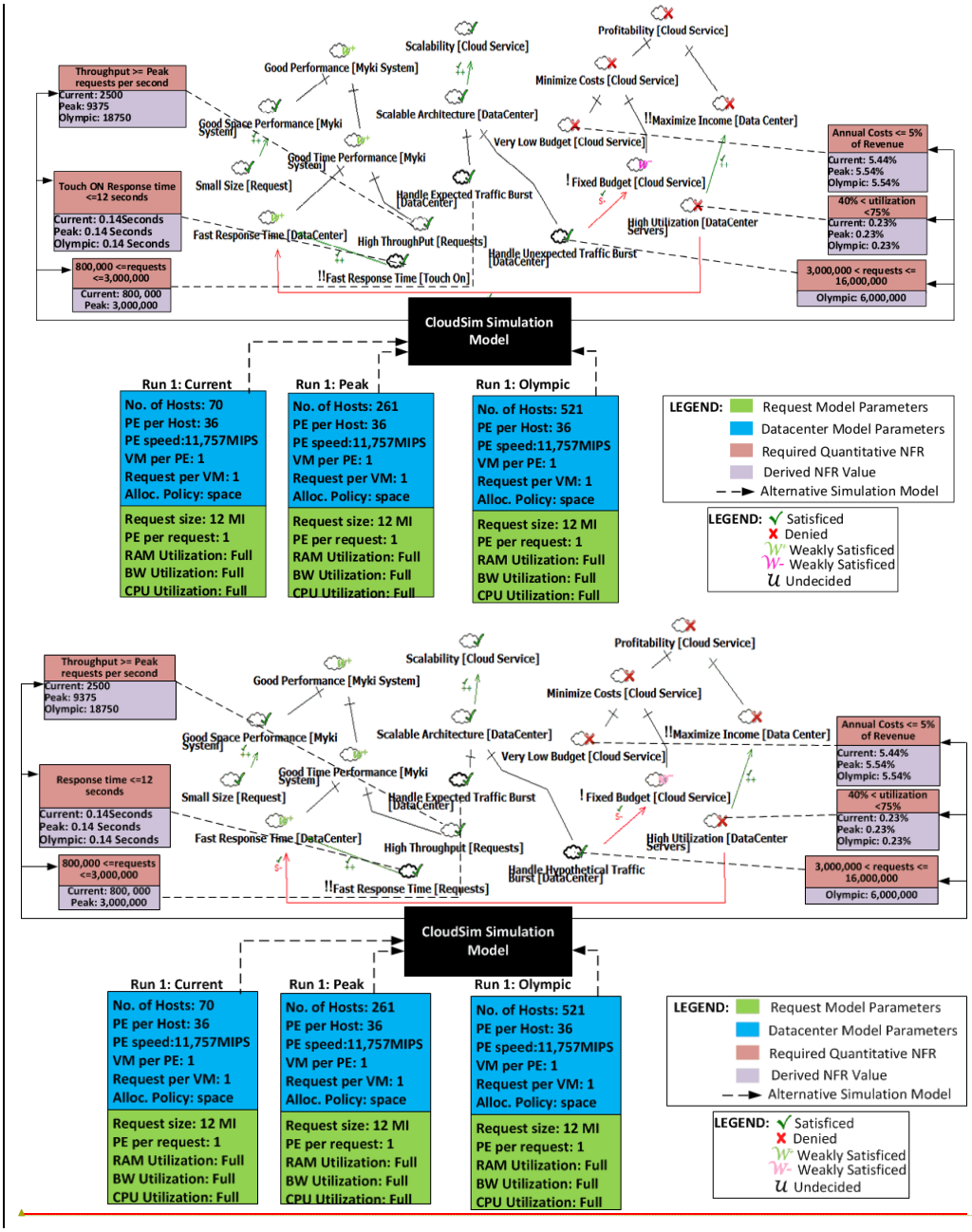


Figure 8: Updating the augmented SIG with Simulation results. Multiple goals are denied while satisficing one critical softgoal

Formatted: Font: Verdana, 10 pt

- Assuming the 3-tiered software architecture style, the initial design for the *Current* workload suggests having 2,500 VMs, each having its own front-end, business logic and database. This is infeasible because while the front end and business logic layers may be horizontally scaled out like this, traditional database management systems do not lend themselves to such scaling out. Hence, the decisions to use one central database, have the VMs communicate with it via a load balancer (as a transaction processing (TP) monitor) and use a second database server for failover. These decisions will be reflected in the final architecture
- The baseline design used a queue length of 1. To improve utilization, let VMs handle 20 requests threads at once.
- The size of each VM is reviewed downwards and the allocation policies are changed from the space-shared policy used in the baseline to a time-shared policy for both cloudlets and VMs. These policies are ~~described~~ defined in CloudSim ~~in~~ [~~43~~26]
- Because of the above improvements, the number of servers used in the processing of requests is reduced from 70 to 25.
- With the changes to the baseline design, the database tier is observed to be the most likely bottle neck. Database architects will have to manage this effectively, perhaps using the right connection pool size, among other things.

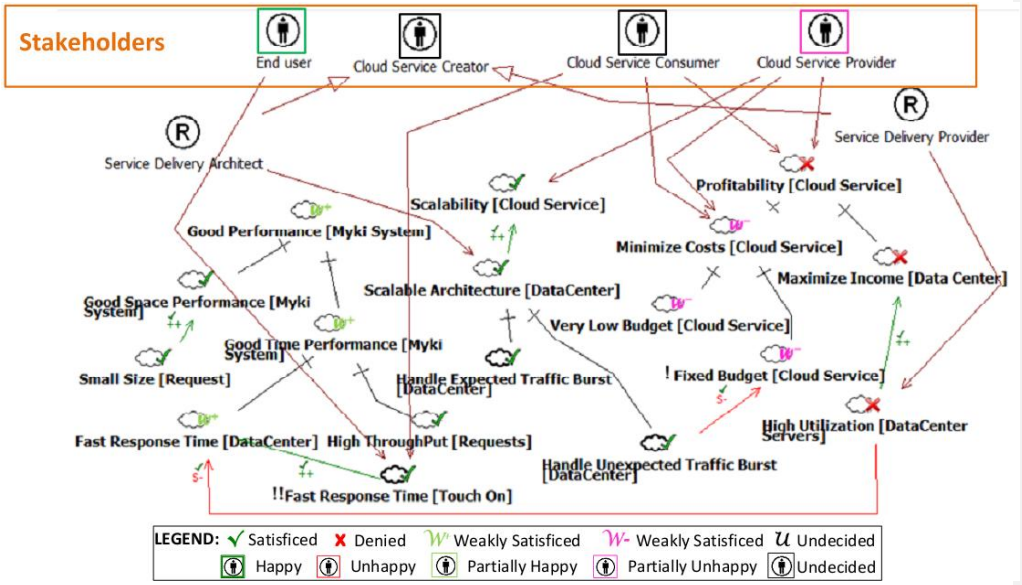
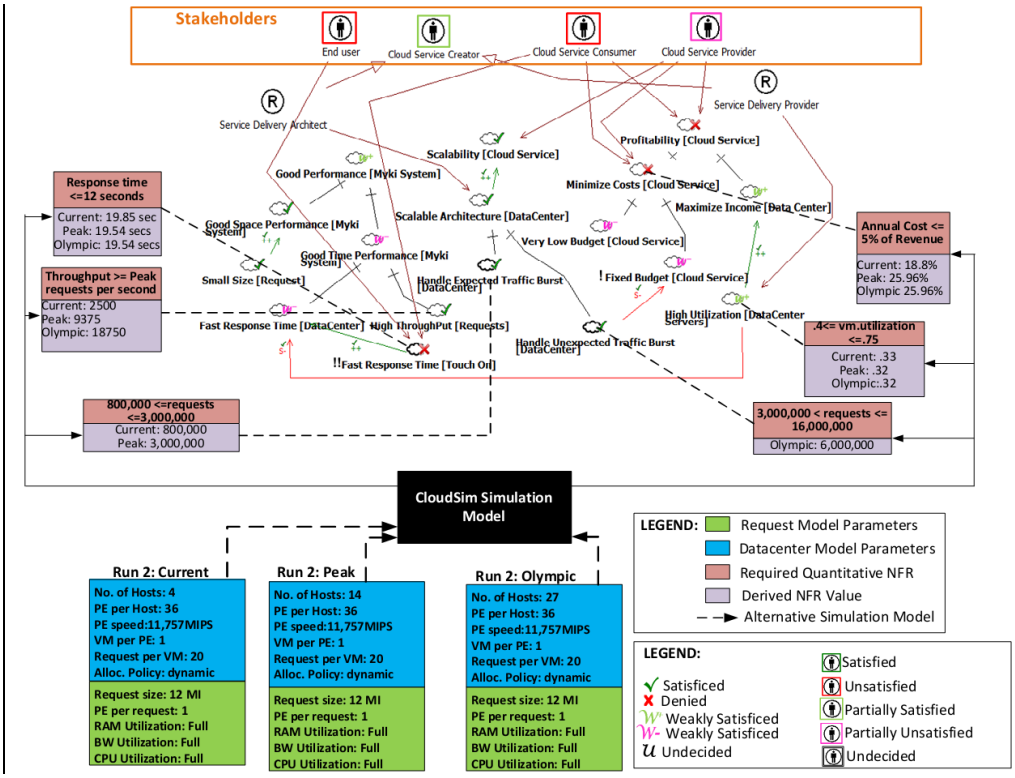


Figure 9: Updated Agent-SIG showing that meeting one critical softgoal satisfies some stakeholders but will likely result in an unfair system for other stakeholder groups.

Similar modifications are made for the *Peak* and *Olympic* workloads and the simulations are re-run. Simulations based on these improvements show some improvements in meeting other stakeholder goals. The results are shown in ~~Figure~~ Fig. 7 as well as the second row in Table ~~5~~ 7 which show that, although we use fewer VMs and Hosts (Servers) and the system is better able to scale, the response time and cost constraints have been violated. ~~Figure~~ Fig. 10 shows updates to the augmented Agent-SIG after running the simulation for the "improved" configuration. Note that ~~Figure~~ Fig. 10 combines the augmented SIG with the Agent-SIG diagrams for brevity.

To improve further, we reduce the number of requests handled per VM from 20 to 10, since it seems that having greater queues consumes more computational resources and leads to the much higher response time and cost factors. We also change the request (cloudlet) allocation policy within CloudSim from *DynamicWorkload* policy to a *time-shared* policy and ~~.re we~~ explore the possibility of allocating more VMs per CPU core. The results of these are shown in the third row of Table ~~5~~ 7 and in Fig. 7. As Fig. 11 shows, these changes result in a design that is likely to better meet the stakeholder goals. Further refinements and optimizations can be carried out, if needed.

The analysis so far has evaluated whether certain configurations are good enough to meet stakeholder goals for a particular kind of workload. More specifically, with respect to the workloads described in Section 3.4.3, while the design so far may cater to the *Current*, *Peak* and *Olympic* workloads, it may not be good enough for the *Australia* workload considering its wider geographical scope and much larger scale. ~~As such, varying workload characteristics will impact upon design choices, in addition to the stakeholders, stakeholder goals and the selected system configuration goals.~~ In the next section, we show how a good enough design for the *Olympic* workload may be derived, assuming the same constraints but with much larger budget and revenues.


Stakeholders
End user
Cloud Service Creator
Cloud Service Consumer
Cloud Service Provider
Service Delivery Architect
Service Delivery Provider
Scalability [Cloud Service]
Profitability [Cloud Service]
Good Performance [Myki System]
Minimize Costs [Cloud Service]
Maximize Income [Data Center]
Response time <=12 seconds
Current: 19.85 sec
Peak: 19.54 secs
Olympic: 19.54 secs
Good Space Performance [Myki System]
Scalable Architecture [DataCenter]
Very Low Budget [Cloud Service]
Annual Cost <= 5% of Revenue
Current: 18.8%
Peak: 25.96%
Olympic 25.96%
Good Time Performance [Myki System]
Throughput >= Peak requests per second
Current: 2500
Peak: 9375
Olympic: 18750
Small Size [Request]
Handle Expected Traffic Burst [DataCenter]
Fixed Budget [Cloud Service]
Fast Response Time [DataCenter]
High ThroughPut [Requests]
High Utilization [DataCenter Servers]
.4<= vm.utilization <=.75
Current: .33
Peak: .32
Olympic:.32
Handle Unexpected Traffic Burst [DataCenter]
!!Fast Response Time [Touch On]
800,000 <=requests <=3,000,000
Current: 800,000
Peak: 3,000,000
3,000,000 < requests <= 16,000,000
Olympic: 6,000,000
CloudSim Simulation Model
LEGEND:
Request Model Parameters
Datacenter Model Parameters
Required Quantitative NFR
Derived NFR Value
Alternative Simulation Model
Run 2: Current
No. of Hosts: 4
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 20
Alloc. Policy: dynamic
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
Run 2: Peak
No. of Hosts: 14
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 20
Alloc. Policy: dynamic
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
Run 2: Olympic
No. of Hosts: 27
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 20
Alloc. Policy: dynamic
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
LEGEND:
Satisficed
Denied
Weakly Satisficed
Weakly Satisficed
Undecided
Satisfied
Unsatisfied
Partially Satisfied
Partially Unsatisfied
Undecided

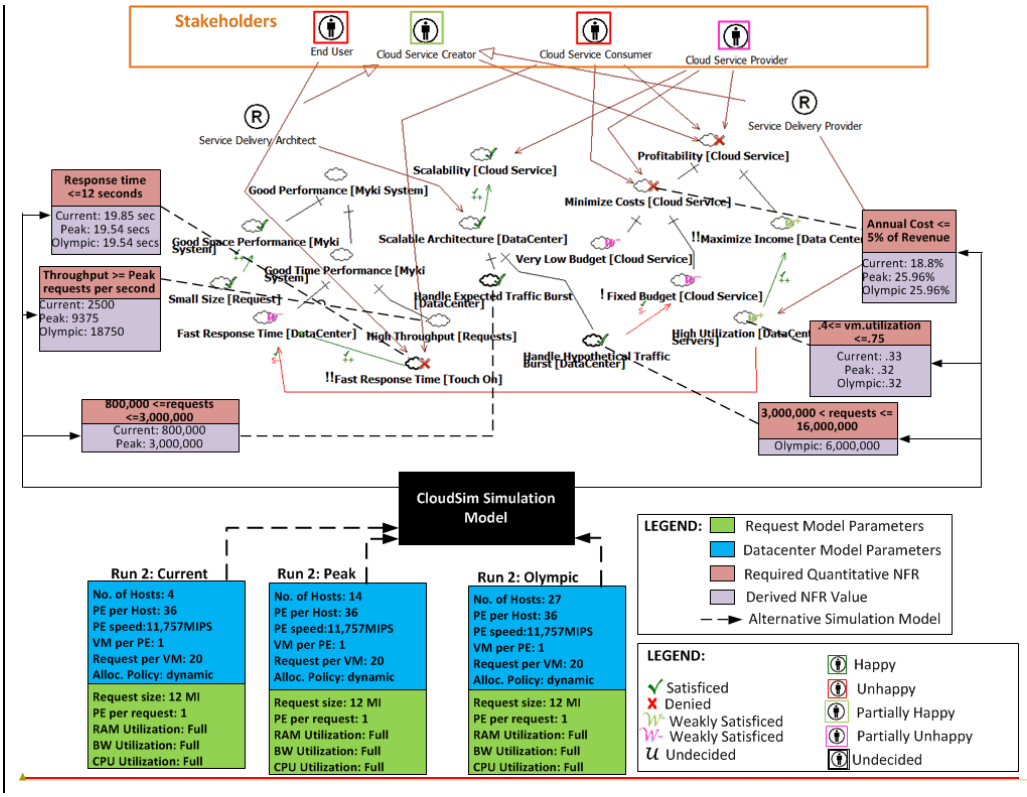


Figure 10: Improving initial design for scalability (utilization) sacrifices performance and cost.

3.5.5 Designing for a different workload type

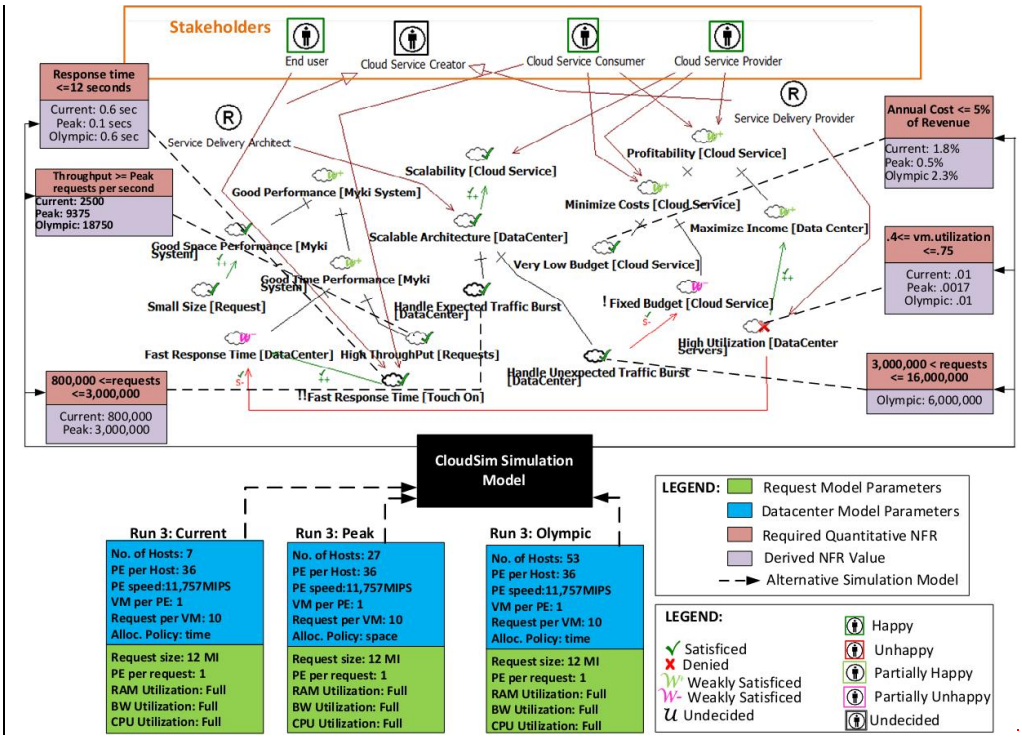
The workloads considered so far shared some similar characteristics which enabled us to consider ~~then~~ them simultaneously. In this section, we consider a workload with significantly different characteristics: the *Australia* workload. This workload results from the hypothetical case that all 6 states in Australia adopt the "myki". Designing for this workload will differ significantly from others considered so far in the following ways:

- The budget and revenue from fare collection will be about 6 times as large as the Melbourne only case (there are 6 states in Australia)
- A choice will have to be made between using a centralized datacenter like the one utilized so far or a set of datacenters distributed around Australia, since distance from datacenter is likely to lead to added performance bottlenecks
- If multiple datacenters are to be used, how many should be used and where should they be located for best performance, given the fixed cost?
- In multiple datacenters are used, network and other costs will be more significant.

Granted that many application level and data storage optimizations will need to me made for best performance in the *Australia* workload, we continue to use the same request model from earlier examples to keep the discussion simple. First we evaluate the cost, scalability and performance tradeoff for a single datacenter configuration and see how well this meets stakeholder goals. Based on the simulations, more datacenters can be considered in the architecture. Keeping with the same assumptions and estimation technique from Sections

Formatted: Font: Verdana, 10 pt

3.4.1.2 and 3.4.1.3, Table ~~6~~ 7 and Fig. 12 shows the results of running the simulation for the *Australia* workload, using the earlier described visual notation.


Stakeholders
End user
Cloud Service Creator
Cloud Service Consumer
Cloud Service Provider
Response time <=12 seconds
Current: 0.6 sec
Peak: 0.1 secs
Olympic: 0.6 sec
Service Delivery Architect
Service Delivery Provider
Annual Cost <= 5% of Revenue
Current: 1.8%
Peak: 0.5%
Olympic 2.3%
Profitability [Cloud Service]
Scalability [Cloud Service]
Throughput >= Peak requests per second
Current: 2500
Peak: 9375
Olympic: 18750
Good Performance [Myki System]
Minimize Costs [Cloud Service]
Maximize Income [Data Center]
Scalable Architecture [DataCenter]
.4<= vm.utilization <=.75
Current: .01
Peak: .0017
Olympic: .01
Good Space Performance [Myki System]
Very Low Budget [Cloud Service]
Good Time Performance [Myki System]
Small Size [Request]
Handle Expected Traffic Burst [DataCenter]
Fixed Budget [Cloud Service]
High Utilization [DataCenter Servers]
Fast Response Time [DataCenter]
High ThroughPut [Requests]
Handle Unexpected Traffic Burst [DataCenter]
800,000 <=requests <=3,000,000
Current: 800,000
Peak: 3,000,000
Fast Response Time [Touch On]
3,000,000 < requests <= 16,000,000
Olympic: 6,000,000
CloudSim Simulation Model
LEGEND:
Request Model Parameters
Datacenter Model Parameters
Required Quantitative NFR
Derived NFR Value
Alternative Simulation Model
Run 3: Current
No. of Hosts: 7
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 10
Alloc. Policy: time
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
Run 3: Peak
No. of Hosts: 27
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 10
Alloc. Policy: space
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
Run 3: Olympic
No. of Hosts: 53
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 10
Alloc. Policy: time
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
LEGEND:
Satisficed
Denied
Weakly Satisficed
Weakly Satisficed
Undecided
Happy
Unhappy
Partially Happy
Partially Unhappy
Undecided

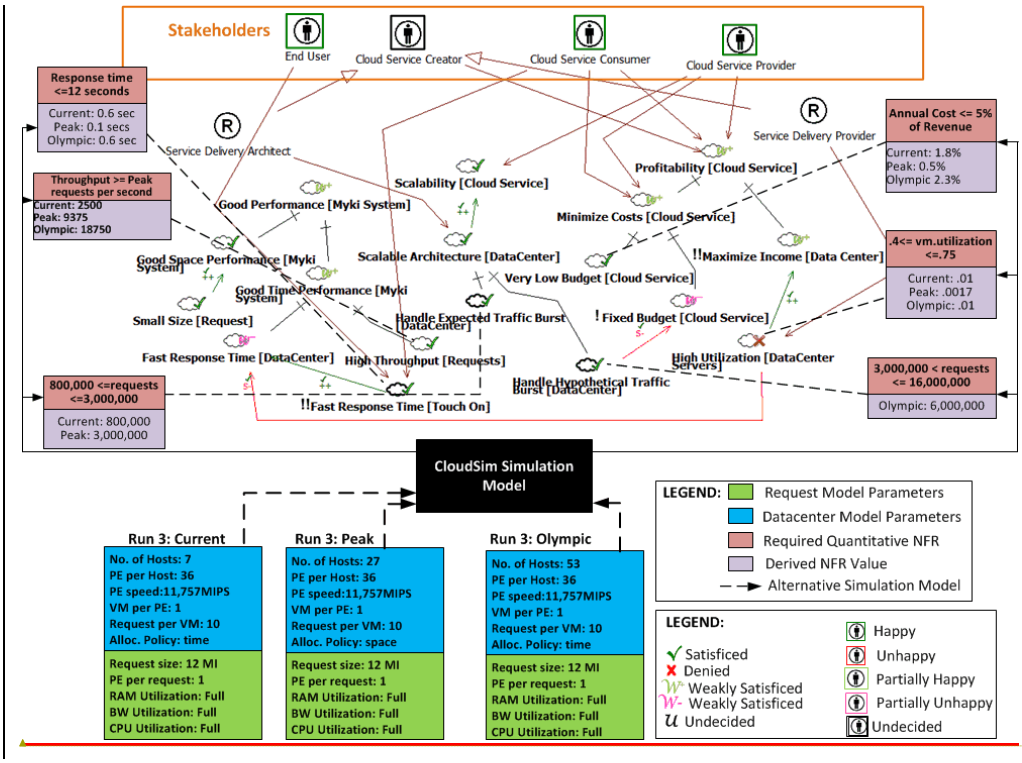


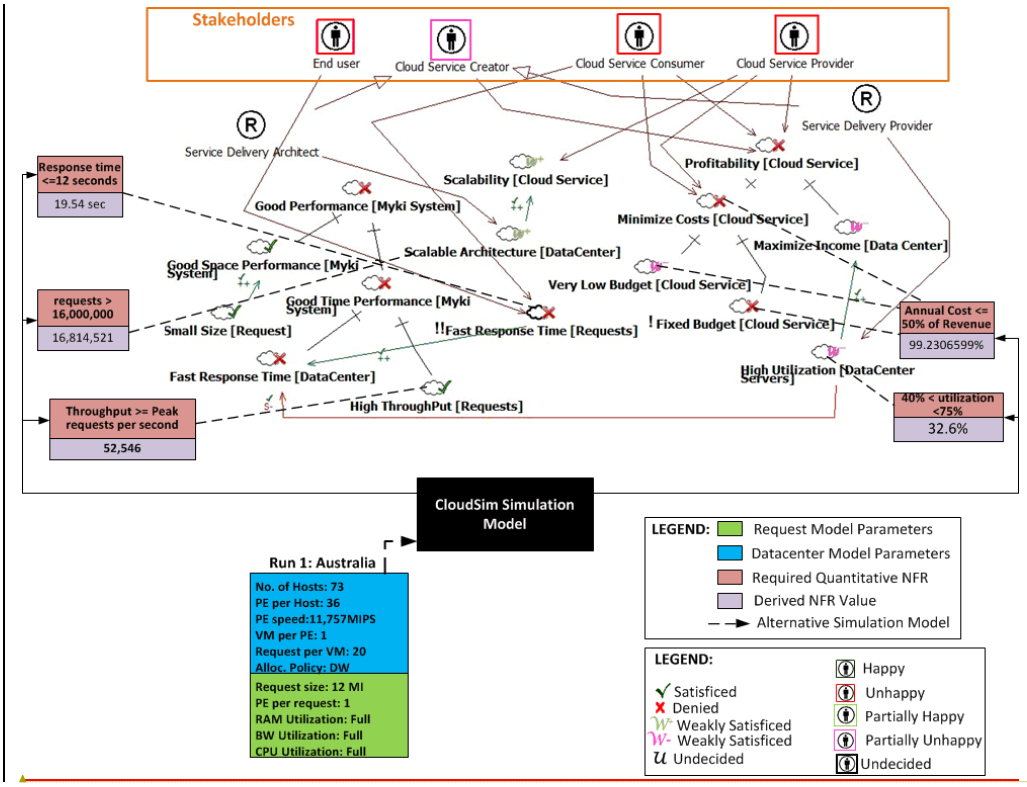
Figure 11: The configurations in the third run seems to satisfice the cost and performance goals simultaneously, while partially satisficing for scalability (poor utilization)

I	Hosts	VMs	RVM	p.time	a.cost p	a.cost h	t.a.cost
1	73	2628	52546	19.54	1,945,142,017.49	1,460,000.00	1,946,602,017.49
2	73	2628	52546	DC1:12.8 DC2: 31.74	1,089,811,117.77	1,460,000.00	1,091,271,117.77

(Abbreviations: **I**="Iteration", **RPM** = "Requests per minute", **Policy** = "Cloudlet Allocation Policy", **DW** = "Dynamic Workload Allocation Policy", **p.time** = "Processing time per request", **p.cost** = "processing cost for all RPM", **a.p.cost** = "Annual Processing Cost", **a.s.cost** = "Annual Server Cost", **t.a.cost** = "Total Annual Coast"

~~**Table 7**~~**Table 8:** Simulation results for the Australia Workload. Annual costs calculated from simulation times.

Formatted: Font: Verdana, 10 pt,



Formatted: Font: Verdana, 10 pt

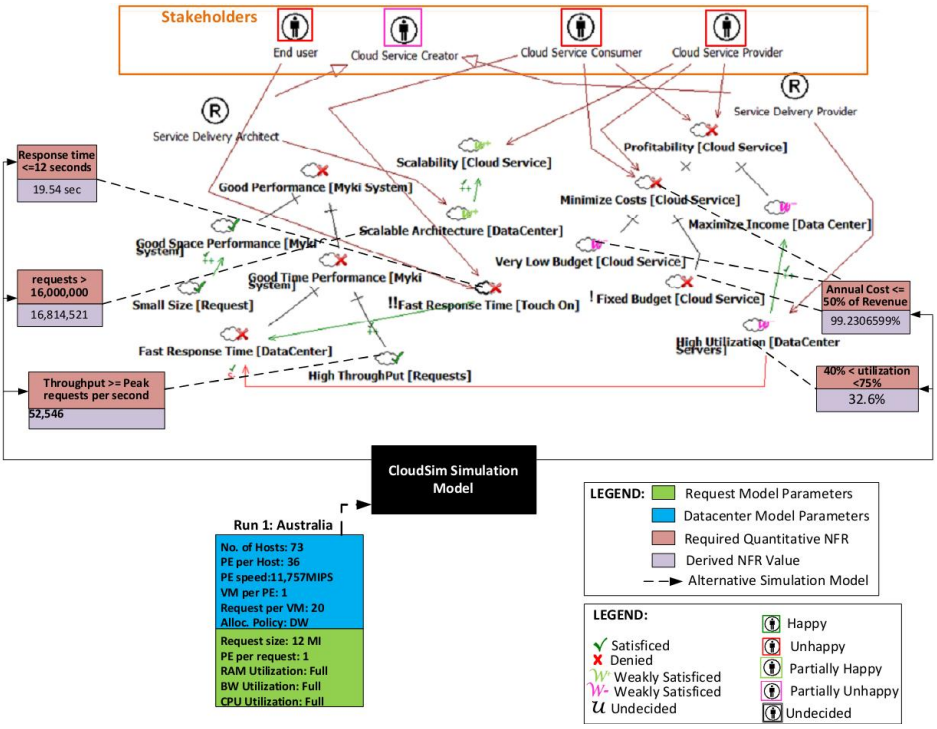


Figure 12: Even with optimizations that worked for the other workloads, using a single datacenter for the *Australia* workload does not meet most of the stakeholder requirements.

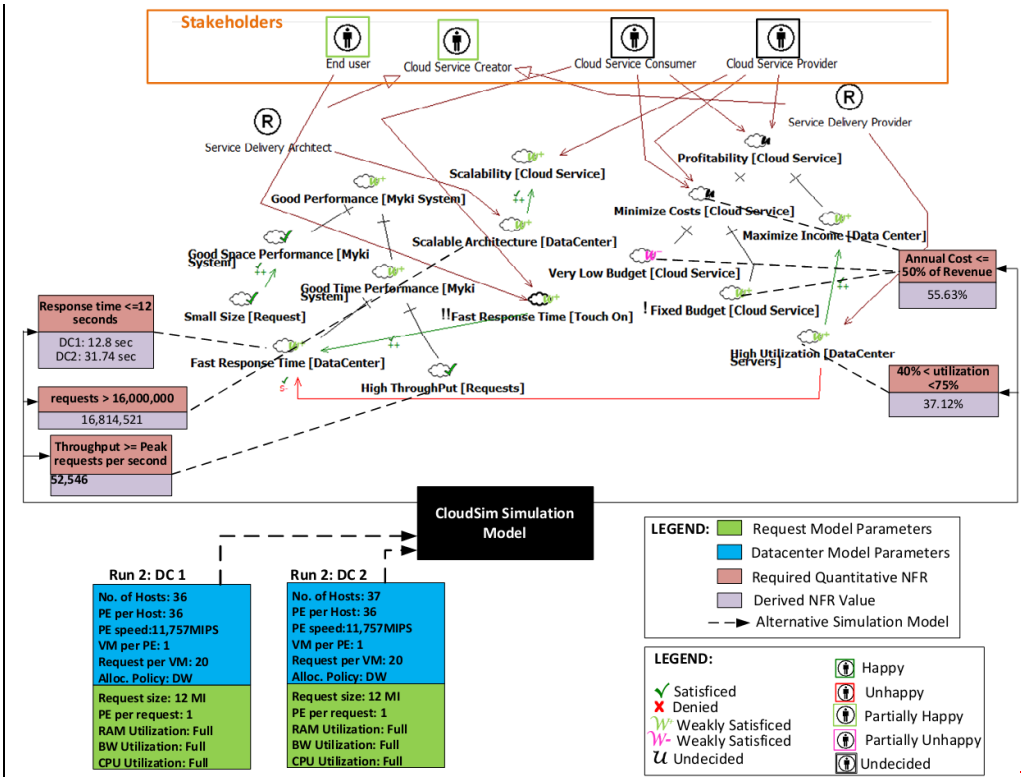
For the second run, we randomly halved the capacity of the datacenter and bandwidth costs used in the first iteration for the *Australia* workload and used 2 such datacenters instead of 1, we observed some interesting results, as shown in Table ~~6~~ 7 and Fig. 13. First, requests sent to the datacenter closest to them had a much less processing time (12.8 seconds) that requests sent to the datacenter farther away(31.74 seconds). Secondly, even though the average processing time for the 2 datacenter scenario (22.27 seconds) was just a little higher than the average processing time in the 1 datacenter scenario, Cloud Computing and Server costs reduced drastically. These results suggest that, with more careful planning, locating bigger datacenters nearer to more densely populated regions reduce costs and improve performance even further, while maintaining good scalability. This seems to confirm ~~the~~ earlier findings ~~in~~ [~~44~~34] ~~which~~ on ~~proposes a tool for investigating~~ the impact of the distance from datacenters on cloud computing costs and performance. ~~and [45] which defines t~~This issue has also been defined as a research problem in Cloud Computing, the solution to which may lead to techniques for ~~by which the optimization~~optimizing ~~between~~ among datacenter location, network performance, costs and other variables ~~may be found~~ [35]. Detailed consideration and treatment of such optimization is part of our future plans.

3.6 Step 5: Translating Derived Model into System Architecture

In this Step, results from the simulation process are translated back into private cloud architecture in the application domain. In Steps 1 and 2 (Sections 3.2 and 3.3) we discussed how to obtain information from the application domain that imposes design constraints on the cloud based system. Steps 3 and 4 (Sections 3.4 and 3.5) how to translate these constraints into the CloudSim simulation model which is then used, as a

proxy for real system architecture, to investigate whether certain configurations will meet the goals of the system. If the configuration is not good enough, changes are made and simulations used to reconfirm their quality, until a satisficing design is obtained.

The simulations in Step 4 have the desired effect that they narrow down the design space from infinitely many designs to a more useful subset. However, it is likely that there will still be a large number of designs in this subset and exploring them all is practically infeasible. One solution to this is to iteratively explore and select among incrementally better alternative designs based on the final simulation model. To do this, one starts with a few candidate designs, perhaps based on the knowledge of creating similar systems in the cloud. These are then evaluated and the one which is most optimal with respect to the tradeoffs in the stakeholder goals is then selected and further improved. Techniques and criteria for selecting among architectural alternatives in Software Engineering ~~are discussed in [46–49]~~have been proposed [36], [37]. In this paper, because of space constraints, we simply show one candidate design and ~~show~~ discuss how architects may improve on it in real-world applications.


Stakeholders
End user
Cloud Service Creator
Cloud Service Consumer
Cloud Service Provider
Service Delivery Architect
Service Delivery Provider
Scalability [Cloud Service]
Profitability [Cloud Service]
Good Performance [Myki System]
Minimize Costs [Cloud Service]
Good Space Performance [Myki System]
Scalable Architecture [DataCenter]
Maximize Income [Data Center]
Very Low Budget [Cloud Service]
Annual Cost <= 50% of Revenue
55.63%
Good Time Performance [Myki System]
Small Size [Request]
!!Fast Response Time [Touch On]
! Fixed Budget [Cloud Service]
Response time <=12 seconds
DC1: 12.8 sec
DC2: 31.74 sec
Fast Response Time [DataCenter]
High Utilization [DataCenter Servers]
40% < utilization <75%
37.12%
High ThroughPut [Requests]
requests > 16,000,000
16,814,521
Throughput >= Peak requests per second
52,546
CloudSim Simulation Model
Run 2: DC 1
No. of Hosts: 36
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 20
Alloc. Policy: DW
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
Run 2: DC 2
No. of Hosts: 37
PE per Host: 36
PE speed:11,757MIPS
VM per PE: 1
Request per VM: 20
Alloc. Policy: DW
Request size: 12 MI
PE per request: 1
RAM Utilization: Full
BW Utilization: Full
CPU Utilization: Full
LEGEND:
Request Model Parameters
Datacenter Model Parameters
Required Quantitative NFR
Derived NFR Value
Alternative Simulation Model
LEGEND:
Satisficed
Denied
Weakly Satisficed
Weakly Satisficed
Undecided
Happy
Unhappy
Partially Happy
Partially Unhappy
Undecided

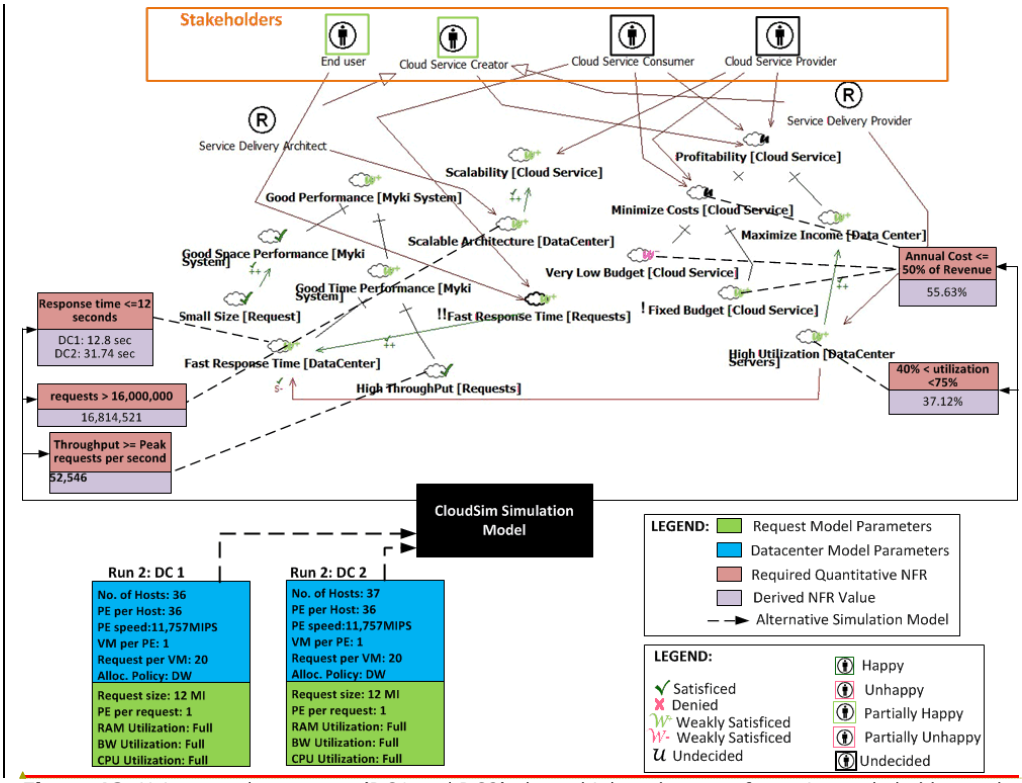


Figure 13: Using two datacenters (DC1 and DC2) shows higher chances of meeting stakeholder goals. Better selection of datacenter locations, among other things, could lead to more improved results.

Formatted: Font: Verdana, 9 pt, Bold

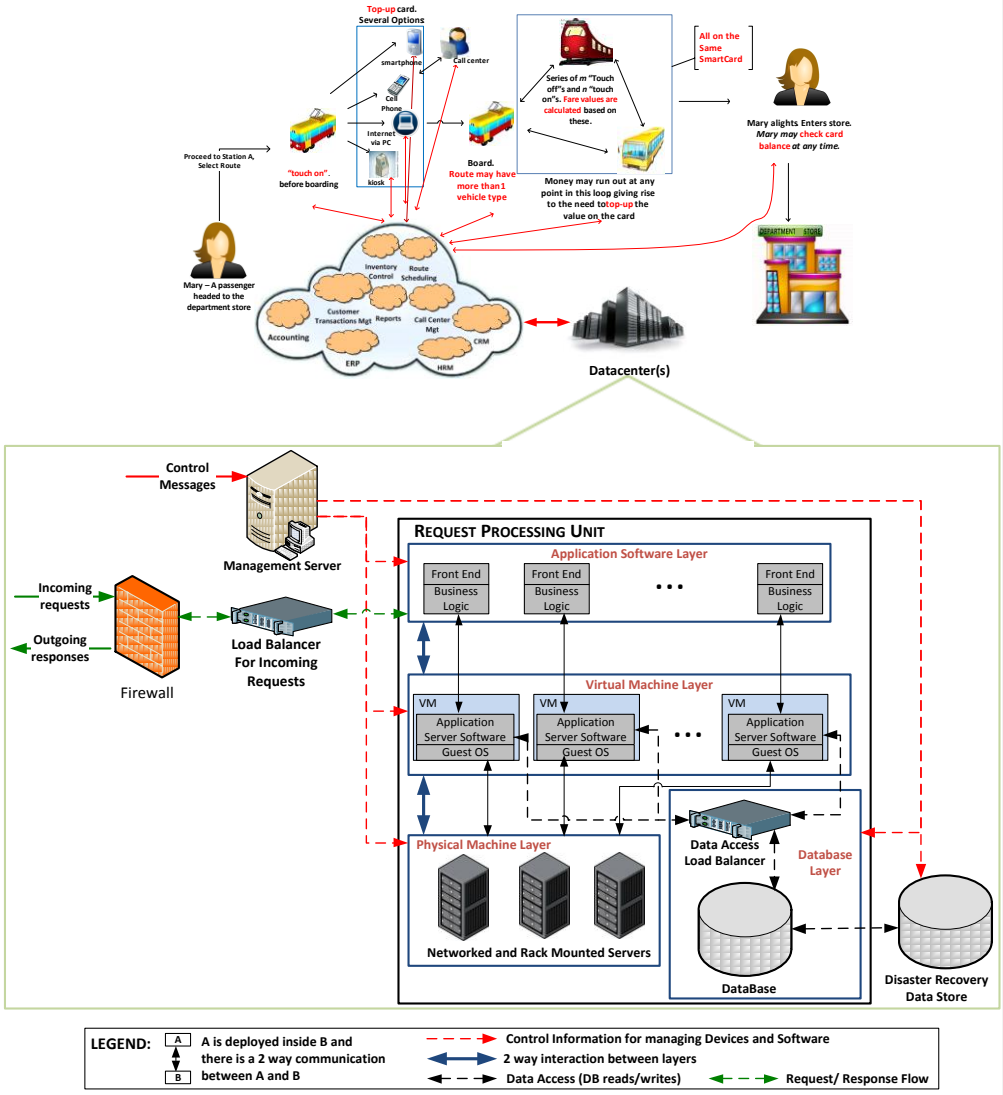


Figure 14: One possible private cloud system architecture for the "myki", based on 3-tier software architecture. Numbers, layout and capacities of devices in the "RPU" are derived directly from simulation for various workloads. Other components added to keep with Private Cloud best practices.

In Fig. 14 we show one such candidate design for the "myki", where the datacenter equivalents of elements from the simulation model are referred to as the "Request Processing Unit (RPU)" for convenience. The numbers and capacities of the devices in the RPU have been omitted from Fig. 14 since these vary from workload to workload. In addition to the cost, scalability and performance goals, the Service Delivery Architect will have to optimize the datacenter architecture for other design goals. For instance, because

the "myki" is a ticketing system, factors such as storage and transfer of customers' sensitive personal (credit card) information are critical considerations. Other likely bottlenecks in the real-time processing of such large volume of requests are (i) the transactional nature of current RDBMS implementations (which makes scalability harder to achieve) and (ii) the fact that third party servers will need to be contacted (e.g. for processing credit card payments) over which the designers of the "myki" have no control. These and other constraints have been identified ~~in~~[~~50~~38] but cannot be directly simulated in CloudSim. Hence, in the real world, the architect will have to account for these and other goals in the system design while still satisfying the system's constraints and meeting the goals of all stakeholders. This is one reason why some devices like firewalls and management servers which were not considered in the simulation model appear as part of the system architecture shown in Fig. 14.

3.7 Step 6: Testing and Real World Experimentation

~~Even though the in~~In the previous five steps, particular care has been taken to consider as many factors as possible. Despite this, mapping from designs obtained from the ideal world of the simulator into real test beds may not be ~~one-to-one~~exact. Some reasons for such divergence of the simulation model from reality include non-functional requirements like security as well as other design considerations which cannot be included as part of the CloudSim Simulation model. ~~We discussed some of these in Section 3.5 for the "myki" example.~~ Hence, before buying or provisioning new hardware for this task, an additional cost cutting measure might be to use one of the numerous available virtual test beds ~~such as~~ [~~51~~39], [~~52~~40]~~, [53], [54]~~. While some of these may come at a cost, the savings from detecting a failure in these test beds are likely to significantly outweigh the cost finding them after implementation and purchasing hardware. Also, in setting up a test bed, ~~[55] provides~~ a large number~~set~~ of industry proven reference architectures for different kinds of cloud-based systems are available for use [41]. This may provide a useful starting point and save costs even further. In the future, we plan to develop strategies for setting up such test beds, both as a way of validating the practical utility of our approach and for providing a rational way of navigating private cloud testing options.

If after setting up the (real or virtual) test beds, the design is found not to match expectations, prior steps in the process might need to be revisited and improved in a new iteration. This is repeated until all stakeholder goals are considered satisficed, perhaps after series of (re)negotiations.

4 DISCUSSION

When compared with other existing techniques ~~(e.g., [11], [56])~~ for using simulation in early stage system development [11], [56], the approach described in this paper provides three key benefits. First, it considers a proper treatment of multiple stakeholders' goals as its starting point and uses this as a frame of reference throughout the entire process, giving rise to higher confidence that the design will be more likely to meet the goals of intended stakeholders. Secondly, it gives more room for human judgment and domain knowledge to guide progress through every phase, providing designers with greater control and flexibility. Lastly, it relies on analysis techniques that are already well known and used in the Software Engineering community, thus the learning curve required to use the approach is expected to be minimal.

Determining the cause and resolution of conflicts in the requirements engineering process for cloud-based systems can be tricky because they could arise solely from the goals, from the stakeholders themselves, from the system's domain or the interplay of these and other factors. This is even more complicated when the multi-dimensional, multi-stakeholder, multi-objective and large scale nature of cloud-based systems is factored in. This work shows one qualitative way of dealing with this interconnectedness of problem sources

towards better resolution of such conflicts while attempting to prevent them from manifesting in the eventually developed system.

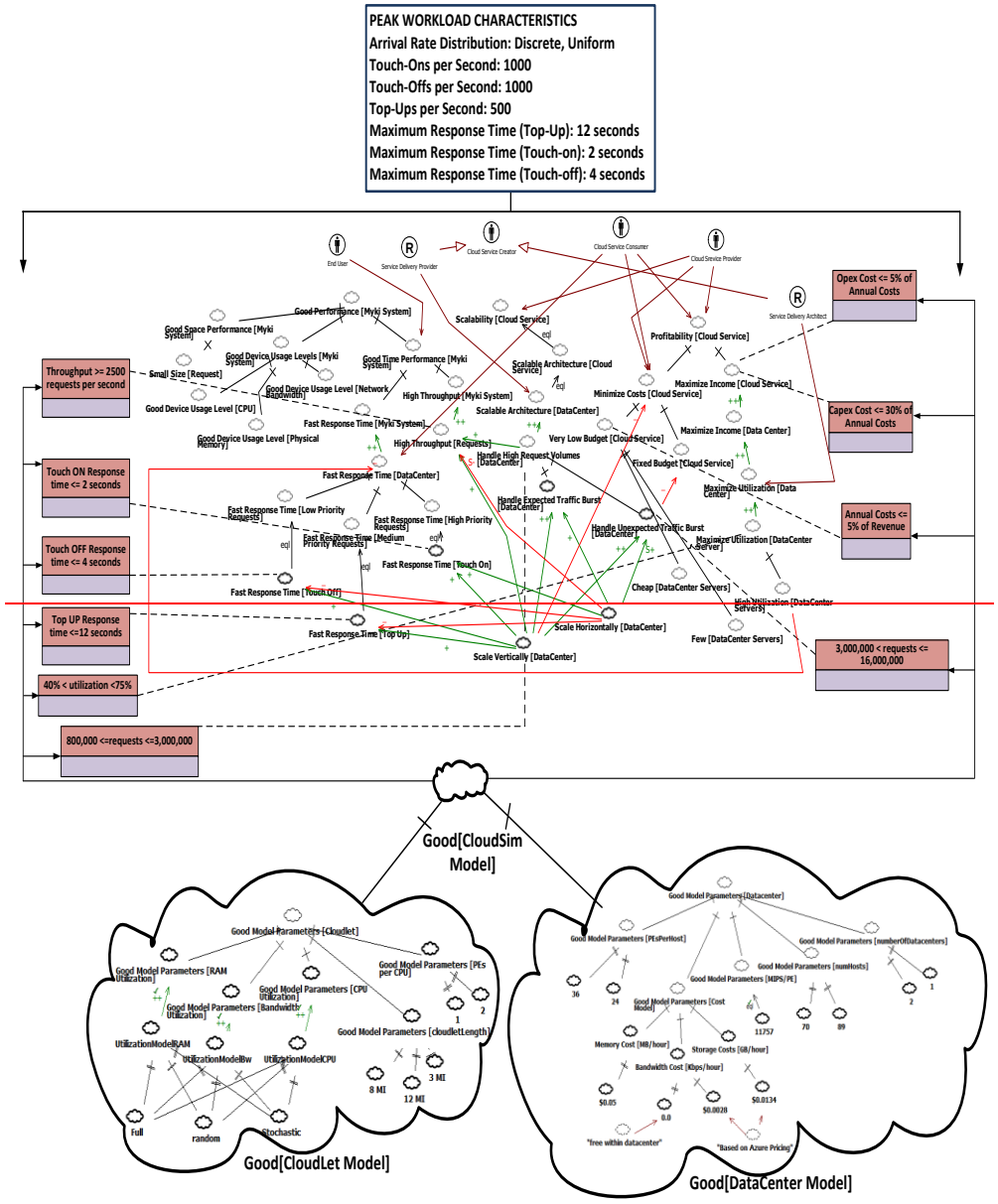


Figure 15 Evaluating the impact of design decisions in the presence of different workload characteristics, multiple stakeholders with different (conflicting) goals is not trivial.

As an aid to the requirements negotiation process, we expect that this work would provide a more accurate decision support than verbal negotiation and conflict resolution since simulations are actually used to provide more objective evidence of how design decisions

affects the goals of each stakeholder in the cloud-based system development process. Another benefit of using simulation in this approach is that since real-world request (inter)arrival rates and other factors are impossible to determine in reality, the designer can evaluate how the system behaves under different known probability distribution functions. This lends~~, lending~~ further credence to our claim that following an approach like the one described in this paper can lead to cloud-based architectures that have a higher degree of confidence in their ability to meet the goals of all interested stakeholders.

Without a systematic approach such as the one described in this paper, evaluating whether cloud-based system architecture will be good enough to meet stakeholder goals could be quite challenging, expensive and time ~~consuming, as the image in Fig. 15 portrays. Consider how architecture for the workloads discussed in this paper could have been done without the use of such a systematic approach. An architect will either have to use intuition based on past experience or some guesstimates in designing the system, with no way of testing if the designs can really meet the goals of all stakeholders until the system is fully implemented, perhaps on a test bed. What if, after implementation, the system is found to not to be satisfactory to some stakeholder goal? In that case, more time and money would need to be spent in (re)developing software and buying/provisioning hardware, assuming the project is not scrapped completely after the first iteration.~~

~~However~~consuming. However, from a science of design perspective, two major concerns that still need to be more properly addressed in order to make this approach more robust are:

i. How can convergence after very few steps be ensured?
ii. Do the incrementally better architectural alternatives considered via the simulation model actually approach the global "optima" in the design space?

Addressing these issues will be important in making sure that the modeling costs involved in ~~the early stages of system development,~~ using the proposed ~~approach,~~approach are not so significant ~~enough~~ as to outweigh the potential time and cost reduction. ~~For now, our approach tends to converge in a number of steps that is linear in the number of high-level stakeholder goals considered initially~~ Although our current results look promising, an essential part of our future work will entail the incorporation of other techniques (e.g., [10]) into our approach which may help ensure convergence after a very few number of iterations.

In Software Engineering, an important part of quantitatively extending qualitative Goal-oriented analysis frameworks has been to formally define sound and complete axiomatization that unambiguously define how reasoning might be done about the quantitative goals. Prior work on this include those which discuss~~Examples in the literature include[57] which discusses~~ quantitative reasoning in the NFR Framework [43], ~~[58] in which the~~ quantitative extension to the NFR Framework ~~is quantitatively extended~~ to aid ~~in~~ requirements traceability [44] and ~~[59] which~~ formal~~ly~~ ~~describes how to~~ reasoning about alternatives in KAOS goal models [45]. In this paper, we gave heuristics for reasoning about levels of goal and stakeholder satisfaction when SIGs are quantitatively extended with design constraints obtained from the domain. Hopefully, this will be a good starting point for the formalizations needed to reason about the quantitative extensions to the ~~augmented~~ SIG as presented in ~~used in~~ this paper.

5 RELATED WORK

A recent goal-oriented simulation technique~~The technique described in~~ [~~11~~10] attempts to fully automate~~s~~ the search through the design space for a design that results in an optimal degree of goal satisfaction.~~for a fixed sample size of the design space~~ This technique ~~and~~ utilizes a multi-objective optimization ~~technique~~ algorithm and ~~to~~ converge~~s~~ on a set of Pareto-optimal solution~~s~~ ~~whose elements~~which approach the global optima within the

sample space considered. This technique, however, requires significant amount of probabilistic modeling and extension of the goal model with numerical values prior to the automation steps. It therefore has the disadvantage that it may not scale very well to very large goal models with a lot of interrelationships since the amount of mathematical modeling required prior to automation ~~is likely~~may ~~to~~ grow exponentially with the size and complexity of the goal model. Also, since the sample of the design space considered is randomly generated and automatically searched, it "robs" human designers of the opportunity to guide the process – something that may be crucial for very critical systems. Our approach allows human architects to guide the process. ~~and w~~We have been experimenting with using ~~the~~ this Genetic Algorithm based technique [10] ~~in [11]~~ complementarily with the one described in this paper, since each one seems to address the other's perceived ~~shortcomings.~~ shortcomings.—It is noteworthy that the inequality expressions in the various constraint symbols (Section 3.5.2) bear a lot of semblance to the objective functions used in [11] to guide the genetic algorithm based search through the design space. We hope to build on this similarity, among others, in creating a hybrid of these two approaches.

Another ~~One work that uses simulation~~approach for ~~to~~ using simulation to evaluate non-functional aspects of cloud-based system designs ~~is [56], in which~~ basically uses results from queuing theory ~~are used~~ to predict whether a cloud-based system with a fixed capacity will remain stable [42]. Stability is measured by the ability of the system to maintain constant response times as more and more requests arrive in the datacenter. However, evaluations are validated against theoretical queuing models such as the M/M/1 model, as opposed to actual stakeholder goals that necessitated such designs in the first place. Considering the multi-stakeholder, multi-objective nature of cloud-based design, it is not clear how such evaluations can be used for improving the system in a rational and fair manner.

In the non-cloud domain, the Architecture Simulation Model (ASM) Approach [~~8~~7–~~10~~9] has been proposed for use in evaluating system design alternatives with respect to specific non-functional goals like performance and scalability in both the design and maintenance phases of system development. The ASM originally proposed the integration of simulation, which is quantitative in nature with goal-orientation, which is qualitative in nature. It therefore served as an important the basis for the work described in this paper.~~It was the ASM which originally proposed the integration of simulation, which is quantitative in nature with goal-orientation, which is qualitative in nature and served as an important the basis for the work described in this paper.~~

Designing a system to be fair to multiple stakeholders is a well-studied topic in the Software Engineering community. The importance of stakeholder identification and analysis in the design process ~~are discussed in~~ [~~22~~16], [~~29~~24], ~~[31] and~~ [~~60~~46] ~~while~~ as well as techniques ~~such as [61–65] have been proposed~~ for dealing with conflicts that arise from the requirements of different stakeholders [47 – 49] have been described. In addition, the topic of negotiating among stakeholders in order to resolve conflicts have also been discussed [50], [~~53~~51]. Integrating more of these results into our approach is likely to make it even more useful.

The work in this paper is related to early stage modeling, analysis and evaluation of system design which have also been well ~~studied. For instance, [69] discusses architecture~~studied. Architecture validation during the analysis phase of system development via the Enterprise Architecture Simulation Environment (EASE) [52] ~~while [70] describes early~~and early stage performance modeling in the design of embedded control software [53] are some of the topics that have received attention in this area. Regarding the use of non-functional requirements for early-stage performance evaluation of designs, ~~[71] discusses~~ an

ontology-aided performance engineering approach using the NFR Framework has been proposed [54].

Although Capacity Planning is expected to be less important in cloud-based systems ~~as~~ than it has been for traditional enterprise systems, it can still be a critical success factor in an organization's Cloud Computing Adoption strategy. To this end, several work on capacity planning for cloud-based systems have appeared ~~(e.g., [72~~55~~–77~~57]). However, these have been mostly focused on datacenter resource optimization techniques with little or no consideration of how decisions to use such techniques can affect the stakeholders of the cloud-based system. To this end, this work can help capacity planners to better integrate actual stakeholder needs into their work, considering the conflicts that may exist among such stakeholders and their goals.

6 CONCLUSION

~~The approach described in this paper demonstrates one way of exploring, evaluating, and selecting among cloud based system design alternatives with respect to stakeholder goals. This is likely to provide better rational decision support and even better cost savings for Cloud Computing, which seems to be among the most critical technological innovations for cost savings, while resulting in architectures that are more likely to be good enough despite the unique nature of cloud based system design and conflicts arising from stakeholders and their goals. Using this approach, architects can more rationally and systematically transition from stakeholder goals to the architectural design of a cloud computing based system. The work also demonstrates the value of goal oriented – i.e. a rational - approach to the science of design, especially when combined in an interleaving manner with simulation, in understanding and conquering the complexity involved in developing a cloud based system.~~The approach proposed in this paper starts with the understanding of stakeholder goals which are subsequently refined and quantitatively extended based on certain domain characteristics. A CloudSim simulation model is then built based on these, as the means of assessing the impact of design choices on the degree to which the various goals are satisficed. After a number of iterations the approach yields a design which may then be tested in a real cloud deployment. Other contributions made by this paper include a new visual notation to help in managing the complexity of the modeling process as well as heuristics for reasoning about the quantitative extensions to the SIG, which is a qualitative goal modeling framework.

~~The approach described in this paper~~Our approach demonstrates one way of exploring, evaluating, and selecting among cloud-based system design alternatives with respect to stakeholder goals. This is likely to provide better rational decision support and even better cost savings for Cloud Computing, which seems to be among the most critical technological innovations for cost savings, while resulting in architectures that are more likely to be good enough despite the unique nature of cloud based system design and conflicts arising from stakeholders and their goals. Using this approach, architects can more rationally and systematically transition from stakeholder goals to the architectural design of a cloud computing-based system. The work also demonstrates the value of goal-oriented – i.e. a rational - approach to the science of design, especially when combined in an interleaving manner with simulation, in understanding and conquering the complexity involved in developing a cloud-based system.

We are currently working on how our approach can be applied to other layers in the XaaS model, in addition to the Infrastructure-as-a-Service (IaaS) layer explored in this paper, including the Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) layers.~~In addition to the Cloud Computing Infrastructure-as-a-Service (IaaS) layer explored in this paper, we are currently looking at how this approach can be applied to other layers in the XaaS model, including the Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS)~~

~~layers.~~ In the future we plan to investigate how to make the approach converge much faster to satisficing architecture, while developing guidelines for ensuring that the design in each iteration is better than the ones in the previous iterations. Additionally, we plan to conduct a case study in which the results obtained from our goal-oriented simulation approach are compared with values obtained from the actual running system, ~~Designs obtained from this approach will be experimentally implemented and tested in a variety of real cloud test beds to see how well the proposed architectures perform,~~ as a way of further validating ~~the~~ our approach. Consideration will also be given to better modeling and simulation of other performance-related stakeholder activities outside the datacenter which may lead to service degradation if not properly addressed.

7 REFERENCES

[1] M. Armbrust, A. D. Joseph, R. H. Katz, and D. A. Patterson, "Above the Clouds: A Berkeley View of Cloud Computing," UC Berkeley, Tech. Rep., No. UCB/EECS-2009-28, 2009

~~[2] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50–55, 2008.~~

[~~3~~2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, vol. 53, no. 6, p. 50, 2009.

~~[4]~~[3] R. Clarke, "User Requirements for Cloud Computing Architecture," in 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 625–630.

[~~5~~4] S. Zardari and R. Bahsoon, "Cloud adoption: a goal-oriented requirements engineering approach," in Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, SECLOUD'11, 2011, pp. 29-35.

[~~6~~5] T. J. Lehman and S. Vajpayee, "We've Looked at Clouds from Both Sides Now," in SRII Global Conference (SRII), 2011 Annual, 2011, pp. 342–348.

[~~7~~6] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in Proceedings. Fifth IEEE International Symposium on Requirements Engineering, 2001, pp. 249-262.

[~~8~~7] T. Hill, S. Supakkul, and L. Chung, "Confirming and Reconfirming Architectural Decisions on Scalability: A Goal-Driven Simulation Approach," in On the Move to Meaningful Internet Systems: OTM 2009 Workshops, 2009, pp. 327–336.

[~~9~~8] T. Hill, "Software maintenance and operations hybrid model: An IT services industry architecture simulation model approach," in Research Challenges in Information Science (RCIS), 2011 Fifth International Conference on, pp. 1–6.

[~~10~~9] T. Hill, S. Supakkul, and L. Chung, "Run-time monitoring of system performance: A goal-oriented and system architecture simulation approach," in Requirements@Run.Time (RE@RunTime), 2010 First International Workshop on, 2010, pp. 31 - 40.

[~~11~~10] W. Heaven and E. Letier, "Simulating and optimising design decisions in quantitative goal models," in 19th IEEE International Requirements Engineering Conference (RE'11), 2011, pp. 79–88.

[~~12~~11] A. Kertész, G. Kecskeméti, and I. Brandic, "Autonomic SLA-aware Service Virtualization for Distributed Systems," in 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2011, pp. 503–510.

[~~13~~12] R. Jeyarani, R. V. Ram, and N. Nagaveni, "Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 585-586, May 2010.

~~[14] V. Emeakaroha and I. Brandic, "SLA-Aware Application Deployment and Resource Allocation in Clouds," in Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, 2011, pp. 298 - 303.~~
[~~15~~13] "Official Home Page of the myki project." [Online]. Available: http://www.myki.com.au/. [Accessed: 31-Oct-2011].
[~~16~~14] A. C. Yoh, H. Iseki, B. D. Taylor, and D. A. King, "Interoperable transit smart card systems: Are we moving too slowly or too quickly?," Transportation Research Record: Journal of the Transportation Research Board, vol. 1986, no. 1, pp. 69–77, 2006.
[~~17~~15] N. Mallat, M. Rossi, V. Tuunainen, and A. Öörni, "An empirical investigation of mobile ticketing service adoption in public transportation," Personal and Ubiquitous Computing, vol. 12, no. 1, pp. 57–65, 2008.
~~[18] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, pp. 1-10, 2008.~~
~~[19] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in 10th IEEE International Conference on High Performance Computing and Communications, pp. 5-13, 2008.~~
~~[20] T. Grandison, E. M. Maximilien, S. Thorpe, and A. Alba, "Towards a Formal Definition of a Computing Cloud," in 6th World Congress on Services, 2010, pp. 191-192.~~
~~[21] L. Youseff, M. Butrico, and D. Da Silva, "Toward a Unified Ontology of Cloud Computing," Grid Computing Environments Workshop, pp. 1-10, 2008.~~
[~~22~~16] H. Sharp, A. Finkelstein, and G. Galal, "Stakeholder identification in the requirements engineering process," in Proc. Tenth International Workshop on Database and Expert Systems Applications, pp. 387–391, 1999.
[~~23~~17] A. Anton, "Goal-based requirements analysis," in Proceedings of IEEE International Requirements Engineering Conference (RE'96), pp. 136-144, 1996.
[~~24~~18] "IBM Cloud Computing Reference Architecture," 23-Jul-2010. [Online]. Available: https://www.ibm.com/developerworks/mydeveloperworks/blogs/c2028fdc-41fe-4493-8257-33a59069fa04/tags/ccra?lang=en. [Accessed: 08-Mar-2012].
[~~25~~19] "NIST Cloud Computing Reference Architecture." [Online]. Available: http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf. [Accessed: 08-Mar-2012].
[~~26~~20] "The CHAOS report." [Online]. Available: http://www.projectsmart.co.uk/docs/chaos-report.pdf. [Accessed: 08-Mar-2012].
[~~27~~21] Lawrence Chung and Julio Cesar Sampaio do Prado Leite, "On non-functional requirements in software engineering," Conceptual modeling: Foundations and Applications, vol. 5, no. 4, pp. 285-294, 2009.
[~~28~~22] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach," Software Engineering, IEEE Transactions on, vol. 18, no. 6, pp. 483–497, 1992.
~~[29] J. Mylopoulos, L. Chung, and E. Yu, "From object-oriented to goal-oriented requirements analysis," Communications of the ACM, vol. 42, no. 1, pp. 31-37, 1999.~~
[~~30~~23] E. S. K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in Proceedings of the Third IEEE International Symposium on Requirements Engineering, pp. 226–235, 1997.
[~~31~~24] A. van Lamsweerde, "Requirements Engineering: From System Goals to UML Models to Software Specifications", John Wiley & Sons, Chichester (2009) ~~A. Dardenne and S. Fickas, "Goal-directed concept acquisition in requirements elicitation," Proceedings of the 6th international workshop on Software Specification and Design, vol. 97403, pp. 14-21, 1991.~~

[3225] E. Yu, "Modelling strategic relationships for process reengineering," in Social Modeling for Requirements Engineering, 2011, pp. 1-15.

[3326] R. Buyya, R. Ranjan, and R.N. Calheiros, "Modelling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities", International Conference on High Performance Computing & Simulation, pp 1-11, 2009.R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," International Conference on High Performance Computing & Simulation, pp. 1-11, 2009.

[34] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of Cloud resources for real-time services," Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science (MGC'09), pp. 1-6, 2009.

[35] A. Beloglazov and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 577-578, 2010.

[3627] A. Nuñez, J. Vázquez-Poletti, A. Caminero, J. Carretero, and I. Llorente, "Design of a New Cloud Computing Simulation Platform," Computational Science and Its Applications-ICCSA 2011, pp. 582–593, 2011.

[3728] D. Kliazovich and P. Bouvry, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," in IEEE Global Telecommunications Conference (GLOBECOM 2010), pp. 1-5, 2010.

[3829] "Point of Sales Systems (POS) - GAO Research." [Online]. Available: http://www.gaoresearch.com/POS/pos.php. [Accessed: 08-Mar-2012].

[3930] J. D. C. Little and S. C. Graves, "Little's Law," Operations Management, pp. 81-100, 2008.

[40] F. Howell and R. Mcnab, "SimJava: A discrete event simulation library for java," in Proceedings of the first International Conference on Web-Based Modeling and Simulation, 1998.

[4131] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, "Profiling and modeling resource usage of virtualized applications," in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 366–387, 2008.

[4232] "TPC-C - Homepage." [Online]. Available: http://www.tpc.org/tpcc/. [Accessed: 08-Mar-2012].

[4333] R. N. Calheiros, R. Ranjan, and C. D. Rose, "Cloudsim: a novel framework for modeling and simulation of cloud computing infrastructures and services," Arxiv preprint arXiv:, pp. 1-9, 2009.

[44][34] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications," 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446-452, 2010.

[4535] A. Greenberg, J. Hamilton, and D. Maltz, "The cost of a cloud: research problems in data center networks," ACM SIGCOMM Computer, vol. 39, no. 1, pp. 68-73, 2008.

[4636] R. Kazman, M. Klein, and M. Barbacci, "The architecture tradeoff analysis method," in Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '98), pp. 68 - 78, 1998.

[47] J. Asundi, R. Kazman and M. Klein, "Using economic considerations to choose among architecture design alternatives," CMU SEI, Tech. Report, No. CMU/SEI 2001 TR-035, December 2001.

[4837] L. Chung, B. A. Nixon, and E. Yu, "Using non-functional requirements to systematically select among alternatives in architectural design," in Proc. 1st Int. Workshop on Architectures for Software Systems, pp. 31–43, 1995.

[49] R. Kazman, L. Bass, and M. Webb, "SAAM: A method for analyzing the properties of software architectures," in Proceedings of 16th International Conference on Software Engineering, (ICSE'94), pp. 81 – 90, 1994.
[5038] "Mapping Applications to the Cloud." [Online]. Available: http://msdn.microsoft.com/en-us/library/dd430340.aspx. [Accessed: 08-Mar-2012].
[5139] A. I. Avetisyan et al., "Open cirrus: A global cloud computing testbed," Computer, vol. 43, no. 4, IEEE, pp. 35–43, 2010.
[52] "Illinois Cloud Computing Testbed." [Online]. Available: http://cloud.cs.illinois.edu/. [Accessed: 10-Mar-2012].
[53] "Open Cloud Consortium | The OCC is a not-for-profit supporting the cloud community by operating cloud infrastructure." [Online]. Available: http://opencloudconsortium.org/. [Accessed: 10-Mar-2012].
[5440] "Cloud computing research testbed | Technical overview." [Online]. Available: http://www.hpl.hp.com/open_innovation/cloud_collaboration/cloud_technical_overview.html. [Accessed: 10-Mar-2012].
[5541] "Intel® Cloud Builders Secure and Efficient Cloud Infrastructure." [Online]. Available: http://www.intel.com/content/www/us/en/cloud-computing/cloud-builders-provide-proven-advice.html. [Accessed: 10-Mar-2012].
[5642] J. Wang and M. N. Huhns, "Using simulations to assess the stability and capacity of cloud computing systems," in Proceedings of the 48th Annual Southeast Regional Conference, 2010, p. 74.
[5743] P. Giorgini, J. Mylopoulos, and E. Nicchiarelli, "Reasoning with goal models," in Proceedings of the 21st Int. Conference of Conceptual Modeling (ER2002), no. October, pp. 167 – 182, 2003.
[5844] J. Cleland-huang, W. Marrero, and Brian Berenbach, "Goal-Centric Traceability?: Using Virtual Plumblines to Maintain Critical Systemic Qualities," IEEE Transactions on Software Engineering, vol. 34, no. 5, pp. 685-699, 2008.
[5945] A. van Lamsweerde, "Reasoning about Alternative Requirements Options," Conceptual modeling: Foundations and Applications, LNCS, vol. 5600, pp. 380 - 397, 2009.
[6046] L. Chung, D. Gross, and E. Yu, "Architectural design to meet stakeholder requirements," Software Architecture. Citeseer, pp. 545–564, 1999.
[6147] A. V. Lamsweerde and R. Darimont, "Managing conflicts in goal-driven requirements engineering," Software Engineering,, vol. 24, no. 11, pp. 908-926, 1998.
[6248] B. Boehm and H. In, "Conflict analysis and negotiation aids for cost-quality requirements", Univ. of Southern California, Tech. Rep., No USC-CSE-99-530, 1999.
[63] J. Lasky, "Conflict Resolution (CORE) for Software Quality Factors," Rome Lab: Griffis Air Force Base, NY, 1993.
[64] B. Boehm and H. In, "Identifying quality requirement conflicts," Software, IEEE, vol. 13, no. 2, pp. 25–35, 1996.
[6549] A. Finkelstein, M. Harman, and S. Mansouri, "'Fairness Analysis in Requirements Assignments," Requirements, pp. 115-124, 2008.
[6650] H. In, D. Olson, and T. Rodgers, "A requirements negotiation model based on multi-criteria analysis," in Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, pp. 312–313, 2001.
[6751] H. In, B. Boehm, T. Rodgers, and M. Deutsch, "Applying WinWin to quality requirements: a case study," in Proceedings of the 23rd international conference on software engineering, 2001, pp. 555–564.
[68] B. Boehm, P. Bose, E. Horowitz, and M. J. Lee, "Software requirements negotiation and renegotiation aids," in Proceedings of the 17th International Conference on Software engineering, 1995, pp. 243–253.
[6952] S. S. Brink, "Enabling Architecture Validation in the Analysis Phase of Developing Enterprise or Complex Systems using Enterprise Architecture Simulation

Environment (EASE)," in Military Communications Conference, (MILCOM'07), 2007, pp. 1–8.
[~~70~~53] S. Wang and K. G. Shin, "Early-stage performance modeling and its application for integrated embedded control software design," Proceedings of the fourth international workshop on Software and performance (WOSP '04), p. 110, 2004.
[~~71~~54] P. P. Sancho, C. Juiz, R. Puigjaner, L. Chung, and N. Subramanian, "An approach to ontology-aided performance engineering through NFR framework," Proceedings of the 6th international workshop on Software and performance (WOSP '07), pp. 125 - 128, 2007.
[~~72~~55] S. Mylavarapu, V. Sukthankar, and P. Banerje, "An optimized capacity planning approach for virtual infrastructure exhibiting stochastic workload," in Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), 2010, pp. 386 - 390.
[~~73~~56] R. Lopes, F. Brasileiro, and P. D. Maciel, "Business-Driven Capacity Planning of a Cloud-Based IT Infrastructure for the Execution Of Web Applications," in 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1-10.
~~[74] C. Loboz, "Cloud Resource Usage: Extreme Distributions Invalidating Traditional Capacity Planning Models," in Proceedings of the 2nd international workshop on Scientific cloud computing (ScienceCloud '11), 2011, pp. 7-14.~~
~~[75] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in Proceeding of the 7th International Conference on Autonomic Computing (ICAC '10), 2010, pp. 11 - 20.~~
~~[76] Y. Chi, H. Moon, H. Hacigümü, and J. Tatemur, "SLA-tree: a framework for efficiently supporting SLA-based decisions in cloud computing," in Proceedings of the 14th International Conference on Extending Database Technology (EDBT/ICDT '11), 2011, pp. 129-140.~~
[~~77~~57] D. A. Menasce and P. Ngo, "Understanding cloud computing: Experimentation and capacity planning," in Computer Measurement Group Conference, 2009.
[~~78~~58] Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)
[59] Duboc, L., Leiter, E., Rosenblum, D., "Systematic Elaboration of Scalability Requirements through Goal-Obstacle Analysis," *Software Engineering, IEEE Transactions on* , vol.PP, no.99, pp.1
[60] C. B. Weinstock and J. B. Goodenough, "On systems scalability," Software Engineering Institute, Technical Note CMU/SEI-2006-TN-012, March 2006, accessed from http://www.sei.cmu.edu/library/abstracts/reports/06tn012.cfm on 14 August 2012.

RESPONSE LETTER

MANUSCRIPT NUMBER: JSS-D-12-00301

The authors do hereby acknowledge, and appreciate, the painstaking extent to which the reviewers have gone to provide us with feedback on our earlier submission. The errors, typos, inconsistencies and technical issues that they spotted, as well as suggestions that they provided us, were very useful in working to try to improve the overall quality of our paper. We thank all the reviewers. We also hope that the revisions that we have now made based on their recommendations, highlighted in this letter as responses to each of their comments, have indeed strengthened our submission, in their esteemed judgment. Kindly find our responses below, in red.

REVIEWER #1

Comment 1

The paper isn't exactly clear on what its main scientific contributions are. This would be useful to clarify. What appears to me to be the main contributions are (1) an extension of the qualitative goal models of the NFR framework to quantitative goals, (2) the development of an interactive simulation technique on those quantitative models, and (3) the application of such simulation to cloud-based systems.

We have now added a paragraph which summarizes some of our main contributions to the Introduction Section, just before the paper outline as follows:

"The main contributions of this paper are as follows:

1. A method of combining goal-oriented requirements engineering techniques with simulation for cloud computing.
2. A technique for complementing the qualitative goal models in the NFR Framework [58] with a quantitative approach.
3. The development of an interactive, iterative and interleaving simulation technique based on this quantitative approach as well as stakeholder needs.
4. A new visual notation, along with heuristics and guidelines for reasoning about the goal models in the presence of the quantitative additions."

Similarly, a new (first) paragraph was added to the Conclusion section, for this (and other reasons too), ending with the sentence,

"Other contributions made by this paper include a new visual notation to help in managing the complexity of the modeling process as well as heuristics for reasoning about the quantitative additions to the SIG, which is a qualitative goal modeling framework".

Comment 2

One of the weaknesses of the paper (related to the above point) is that it lacks conceptualization. The paper presentation is very much example-driven (which is good to explain concepts) but the underlying fundamental concepts are left for the readers to discover. Sections 3.4 and 3.5 could be improved by making the fundamental concepts used in the approach explicit and giving them precise definitions. For example, these sections seem to refer

to quantitative goals, performance variables, assumptions, design constraints that all seem fundamental but not explicitly defined.

We realize that this could have made things a bit confusing to potential readers, if not corrected. The paper has now been revised to include definitions of the terms mentioned by the reviewer, in the sense that we have used them in Section 3.3, as follows:

Quantitative goals: earlier references to this term were changed to "design constraints", as explained below, except where we are referring to other works which use this terminology.

Assumptions: The heading for Section 3.3.1.2 has been changed from "Assumptions" to "Some Assumptions made in using Little's Law for the "myki", to show that the section contains assumptions we have made in using Little's Law. More explanation for the necessity of such assumptions has also been given in Section 3.3.1.1, paragraph 1.

Design constraints: We have now explained the sense in which we use this term in the first two sentences of Section 3.3:

> In order to use simulation, which is quantitative … it is necessary to augment the qualitative goal model … with numbers that guide decision making. This entails the translation of stakeholders' requirements from the application domain into numerical target values that the eventual system design is expected to meet or exceed …We refer to these target values as "design constraints"

Performance variables: In Section 3.3, we have now explained what we mean by this term. For instance, the third sentence in paragraph 2, Section 3.3.1 asks, "how do we derive these quantities like number of simultaneous transactions, maximum processing time per request, and total number of requests per day which, among others, form the key *performance variables* that are of interest to stakeholders?" The sources, necessity and estimation of these performance variables are dealt with in the rest of Section 3.3.

In addition, much of the introduction to Section 3.3 has been re-written in an attempt to better conceptualize other essential concepts used in the rest of the section. Finally, the process diagram has also been abstracted and conceptualized without details from the "myki" case study, as shown in Fig. 5 below.

Comment 3

In the description of step 3 (Section 3.3), it isn't clear whether the qualitative goal model produced in step 2 is used to guide the elaboration of the quantitative models. Could you clarify this point? And if the qualitative goal model is used to guide the elaboration of the quantitative ones, could you clarify how?

A new paragraph has been added to the introductory portion of Section 3.3 to further clarify the nature of Step 3 activities and to summarize the roles that the qualitative goal model from step 2 plays both in Step 3 and in the rest of the paper, like so:

> "It is important to point out that the aim of this step is not to replace any of the reasoning or elaboration activities and techniques normally associated with goal models. Neither is it meant to be a proposal for a new kind of quantitative goal model. Rather, it complements existing techniques in order to further facilitate the use of goal-orientation with simulation. The utility of the SIG produced in Step 2 to the activities in Step 3 includes:

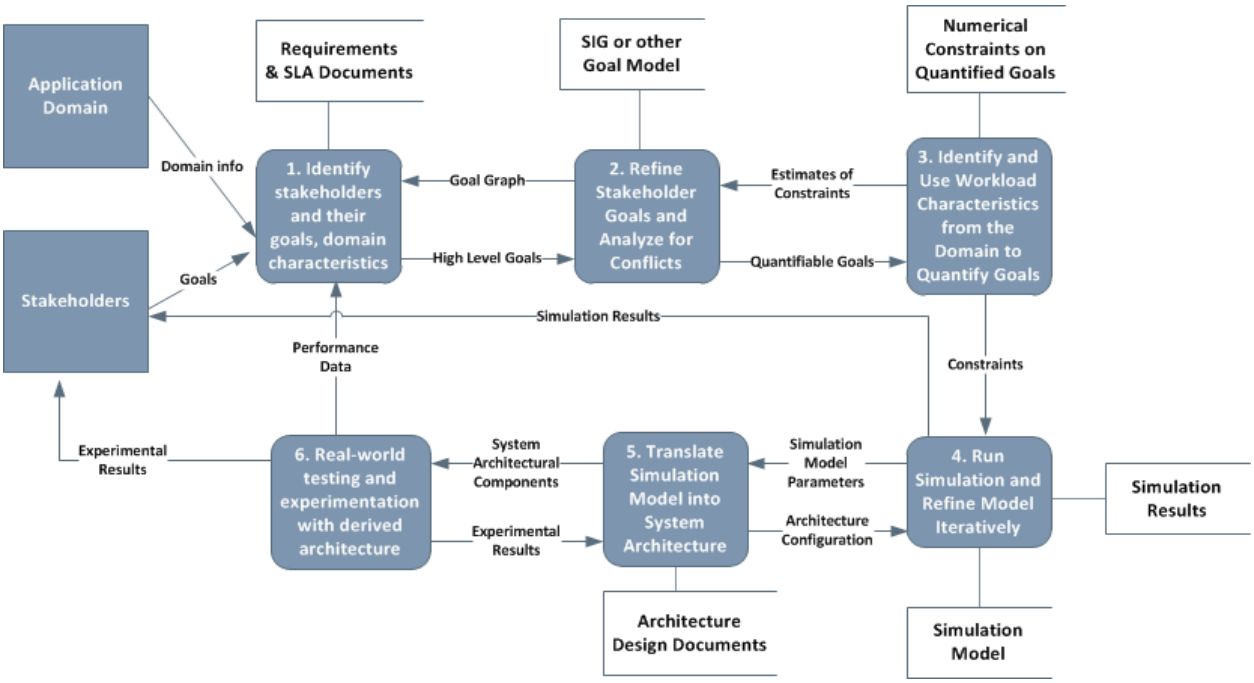


Figure 5: A summary of our 6-step process. All steps are described in the rest of this section.

1. The SIG can help stakeholders negotiate and agree on target values that define what it means for the goals in the SIG to be considered good enough.
2. It helps requirements engineers to discover what domain specific phenomena need to be quantified and fed into the simulation model.
3. It serves as the basis of reasoning about the degree to which the softgoals have been satisficed"

We currently show how this may be done for higher level goals (cost, scalability and performance) and are working on showing how the lower level goals may be quantified. All these, this, in addition to more references to the qualitative goal model in the rest of Section 3.3, should hopefully clarify the point raised by the reviewer.

Comment 4

The elaboration of the scalability requirements and quantitative models appear to be pretty crude (a real scalability assessment would need to consider a wider range of variable than just the maximum number of requests *per day*). Recent work on goal-oriented elaboration may help you refine the models (Duboc et al., Systematic Elaboration of Scalability Requirements through Goal-Obstacle Analysis, TSE 2012)

Yes, the recent work on goal-oriented elaboration of scalability goals was helpful, both in conceptualizing our approach and for explaining some of the choices we had made earlier. We would like to thank the reviewer for pointing us in this direction. Also, we have now referenced this work, as such. For instance, in sentence 1, Section 3.3.2, we adopt directly from the suggested paper in defining what a scalability goal is:

> The concept of a scalability goal has recently been more precisely defined in terms of specified scaling assumptions [59], wherein a *scalability goal* is

> *"A goal whose definition and required levels of goal satisfaction make explicit reference to one or more scaling assumptions".*

In turn, a *scaling assumption* is

> *"A domain assumption specifying how certain characteristics in the application domain are expected to vary over time and across deployment environments".*

In addition, we have also added a paragraph to discuss the issue of whether other goals are expected to remain fixed or variable in their target values as the system is scaled to handle larger and larger workloads. To this end, the last paragraph in section 3.3.2 now reads:

> "One factor that needs to be accounted for is whether other system goals are expected to have fixed or changing target values as the workload varies. Cases where system goals are expected to remain fixed have been defined as *scalability goals with fixed objectives* (SGFO) while those in which the system goals can vary have been defined as *scalability goals with varying objectives* (SGVO) [59][1]. For the "myki", one SGFO is that the system should be …"

We have however chosen to limit our treatment of scalability goals to these points, as well as workload related scalability concerns. We feel that this level of complexity in the elaboration of scalability requirements is quite sufficient for our purposes, and appropriate, based on the definition of scalability as "an ability to support increased workloads with adequate service levels or performance" [Menasce, D. and Vigilio, A.: Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning. Prentice Hall PTR, Englewood Cliffs (2000) – reference [61] in the latest version]. We have also added references to other publications where more refined treatment of scalability may be found, including the recent work recommended by the reviewer.

Comment 5

The simulation approach in Section 3.5 seems to be about estimating the impact of very low-level design decisions related to the setting of parameters for the virtual machines. Shouldn't goal-based analysis also involve design decisions at a higher-level such as considering alternative goal refinements, agents, and functionalities for the system? The paper could at least be clearer about what kinds of design decisions this approach is intended for.

We agree and acknowledge that goal models are used for the kinds of analyses mentioned by the reviewer. However, one other thing that goal models are used for is to better deal with nonfunctional requirements that drive the system design and this is the focus of our paper, with respect to cloud based systems. We do not make any assertion in the paper that this is the only thing that goal models may be used for. In addition, we have now tried to make clear in a number of places in the paper that our approach aims to use simulation for evaluating the impact of Cloud-Computing infrastructural design choices on the proposed system's ability to meet the needs of its stakeholders. Kindly see the last sentence in the 4th paragraph in the Introduction:

> "Our use of simulation is for investigating the impact of Cloud-Computing infrastructural design choices on the proposed system's ability to meet the needs of its stakeholders"

[1] The abbreviations are not part of the original definitions, but have rather been used for brevity in the rest of the paper.

as well as the first paragraph in page 8:

> "In this paper, we use Cloudsim to investigate the impact of design choices made in the datacenter configuration on concerns at a higher level of abstraction, closer to stakeholders' goals…"

Comment 6 (Typos)

- In Section 1, 1st paragraph: "such cost reduction" -> "such as cost reduction"

This typo has now been fixed to read, "such as cost reduction".

- In Section 2.3, the paragraphs on cloud service creator and cloud service provider seem to be mixed up.

This typo has now been fixed by swapping the definitions of cloud service creator and cloud service provider.

- Section 3.5.4, 1st paragraph. "the simulation model will need to be improved". I suppose you mean "the system design will need to be improved".

The ambiguity resulting from this has now been clarified by re-writing the sentence as follows:

> "Hence, the system design, as captured in the simulation model, will need to be improved towards meeting the goals of as many more stakeholders as possible"

- at the end of Section 3, "3.5 Step 6: " should be "3.7 Step 6".

This typo has now been fixed.

REVIEWER #2:

Comment 1

The authors extrapolate on a real-world system for illustrating their approach. Many assumptions are made regarding the application domain and system architecture, some of them arguable. Yet, such assumptions are unavoidable and authors provide the rationale for most numbers. I find the example sufficient and adequate for illustrating the proposed solution. In future work, I would like to see a case study in which the results of the simulation and goal satisfaction are compared with the running system.

This is a very valuable suggestion, one that we had already included in the original manuscript, in the last sentence of the first paragraph in step 6:

> "In the future, we plan to develop strategies for setting up such test beds, both as a way of validating the practical utility of our approach and for providing a rational way of navigating private cloud testing options."

and also as part of the "Future Work" portion of Section 6:

> "Additionally, we plan to conduct a case study in which the results obtained from our goal-oriented simulation approach are compared with values obtained from the actual running system, as a way of further validating our approach."

We realize that our earlier choice of words in the latter may have somewhat obfuscated our intentions and the above statement is part of our efforts to re-write that, in an attempt to clarify.

Comment 2

How exactly the quantities derived from the simulations are used to determine the various degrees of satisfaction of goals? Step iv of Section 3.5.2 (page 14) mentions that claim goals are used for this purpose. This is an important part of the process and I would like to see it clearly spelled out.

We have now added more rules, set forth in Table 5 and exemplified in item (vi) page 16, to the heuristics for reasoning about how the quantities obtained from simulation can help determine the various degrees of goal satisfaction. The new rules cover most cases that are expected to be common in the way that constraints on the various quantities are specified.

We also added one limitation that we foresee in the application of the heuristics, as something which needs to be researched further in the future:

> "One issue concerns the implicit assumption we have made in the describing the visual notation, viz.: constraint symbols can only link to one softgoal. What if multiple constraints link to the same goal, having different levels of satisfaction? … This is open to further research"

Comment 3

I am convinced of the relevance and contribution of the work, however there are two claims towards the end of the paper that I find unsound. On page 26, the authors say that their "approach tend to converge in a number of steps that is linear in the number to the number of high-level stakeholder goals considered initially". No data is presented to support this claim. On page 27, authors say that their solution are likely to provide "better cost savings for Cloud Computing", yet the authors themselves say that the upfront modeling costs must be addressed and analyzed in future research.

For the first "claim", the point was that although our initial results look promising, we are yet to find a good way to guarantee that the number of iterations needed to make the approach converge towards an architecture that is likely to be good enough for all stakeholders. We have now rewritten the paragraph to reflect this, while pointing out aspects that still remain to addressed in the future:

> "Without a systematic approach such as the one described in this paper, evaluating whether cloud-based system architecture will be good enough to meet stakeholder goals could be quite challenging, expensive and time consuming. For example, this may involve purchasing as well as manually/physically configuring a cloud and/or testing it. However, from a science of design perspective, two major concerns that still need to be more properly addressed in order to make this approach more robust are:
>
> i. How can convergence after very few steps be ensured?
> ii. Do the incrementally better architectural alternatives considered via the simulation model actually approach the global "optima" in the design space?
>
> Addressing these issues will be important in making sure that the modeling costs involved in using the proposed approach does not get so significant as to outweigh the potential time and cost reduction. Although our current results look promising, one essential part of our future work will entail the incorporation of other techniques … into our approach which may help ensure quick convergence. Providing tool

support for our approach is also another avenue by which modeling costs may be further reduced, and one that we are also considering, going forward."

We hope this also clarifies the second point and would like to thank the reviewer for spotting and raising this critical issue.

Comment 4

I respect the authors' view, however I can't help saying that I fundamentally disagree that an architecture, to be considered scalable, must handle unforeseen spikes (page 11). Such goal is by nature unachievable. It is duty of the requirement analysts to characterize the variation on the application domain to the best of their ability. The design of a system should be guided by quantitative and documented assumptions on the application domain variations.

Having taken a second look, based on the reviewer's reasonable and very well stated objections, we have removed the references to requirements that the architecture must be able to handle "unforeseen spikes" to be considered scalable. All the SIGs in the paper have also been edited to reflect this. In addition to this, we have now incorporated, from recently published work, some concepts that should, hopefully, convey that our ideas about scalability are grounded in results from deeper studies of the subject.

Comment 5

- Page 6, Figure 3 is not consistent with text. Discussion on the previous page says that Cloud Service Provider may comprise of two roles, Service Development Architect and Service Deliver Architect. However, Figure 3 shows these as roles of the Cloud Service Creator.

This inconsistency has now been resolved by correcting the typo in the discussion section that the image represents.

Comment 6

- Page 6, 1st paragraph: I suggest Lamsweerde's book as a more up-to-date reference for the KAOS method (Requirements Engineering: From System Goals to UML Models to Software Specifications)

We have now updated the reference for the KAOS method to the more recent one as suggested.

Comment 7

- Page 8, Section 3: Having Figure 5 as the very first thing on the section is a bit confusing. I suggest moving Figure 5 to after Section 3.1. In the same figure, why step 6 has a different colour?

Figure 5 has now been redrawn entirely (as shown in one of our responses to Reviewer 1) and moved to the end of the introductory portion of Section 3. An explanation has been added to the same section as to why Step 6 initially had a different color – the experimental results reported in the paper do not cover the activities in step 6, which is part of our future work. All steps in the new diagram now have the same color.

Comment 8

- Page 10, 1st paragraph: which operationalization context are being considered for the volume of requests? Touch on, Touch off, Top up, all?

Although a more refined treatment (i.e. one which considers the different types of requests) might be more desirable in practice, we used an aggregate of all operationalization contexts in the paper, for illustration and considering space constraints. We added a third paragraph to explain this, as well as the tradeoffs involved, to Section 3.3.1 – the section containing the paragraph that the reviewer's question is based on:

> "Realistically, a more refined treatment might be desirable in identifying and deriving numerical values for the related performance variables. For instance, instead of computing the processing times or number of requests per day for all requests as an aggregate like we do in illustrating our ideas in this paper, these values may be computed for the individual types of request (i.e. touch on, touch off, top-up). All the estimation techniques discussed in the rest of this section will remain applicable, and the increase in the amount of computations is expected to be offset by the benefits to performance engineers in terms of a more detailed view of the system under investigation."

Comment 9

- Page 11, 1st paragraph of Section 3.4.2: Where the 3 million rpd at peak capacity comes from?

This Section (now 3.3.2) has now been re-written in its entirety. Specifically, sentence 2, third paragraph in the new Section 3.3.2 addresses the reviewer's comments by pointing to the source of data used for creating all the scaling assumptions. 3 million rpd at peak capacity came from our interactions with the "myki" operators although we have attempted to obfuscate the actual numbers a bit:

> "…The values used for the *Current* and *Peak* are somewhat obfuscated from real data obtained from "myki" operators."

Comment 10

- Page 12, 1st paragraph of Section 3.5.1: I suggest moving the CloudSim authors explanation to the background section.

The CloudSim authors' explanations have been moved as suggested.

Comment 11

- Page 18, last paragraph: confusing phrase --- As such, varying workload characteristics will., in addition to the stakeholders,."

The entire sentence has been removed, for clarity and also because the point it was intended for has been made elsewhere in the paper.

Comment 12

- Reference [78] doesn't seem to be cited.

The reference has now been properly cited as the book, "Non-Functional Requirements in Software Engineering" (Chung et al). See Page 2, last paragraph and Section 3.4.2, item (iv), the penultimate paragraph in Section 1 and the first paragraph in Section 2.4. Please note that this is now Reference [58] due to the overall reduction in the number of references.

Comment 13

- Sometimes authors refer to "Fig. X" and sometimes to "Fig X" (without the dot)

All short forms have now been cross-checked to be of the form Fig. X

Comment 14

- The paper is lengthy, but I understand there are a number of graphs and a detailed process to explain. Here are some suggestions where the paper might be reduced:

-- Page 4, Section 2.2: The authors understanding of cloud computing would be enough.

Section 2.2 has been reduced to our understanding of cloud computing.

Comment 15

-- Page 12, Section 3.5.1.1: Is it necessary a section to justify why the authors choose and interleaved method rather than an incremental? Perhaps this could be more briefly mentioned in the introduction of Section 3.5.

We originally included this paragraph because, although it made more pedagogic sense to present the approach sequentially like we did in the paper, the reader may be lulled into thinking that the process was meant to be followed in such strict sequence. What we have now done is to take out the old Section 3.5.1.1 completely since the introduction to our 6 step approach (Section 3, overview portion) and a few other places in the paper already discussed the interactive, iterative and interleaving fashion in which the approach is designed to be used. For example, the first sentence in the second paragraph in Section 3 now reads:

> "The Gane-Sarson DFD in Fig. 5 shows the 6 steps, which are not meant to be followed in strict sequence. Rather, they are envisioned for use in an interactive, interleaving and iterative manner whereby …"

Also, the final paragraph in the Discussion section now reads:

> "Finally, we wish to re-emphasize that, in this paper, we presented the 6 steps in our approach in sequence purely for pedagogic reasons. Such a purely incremental/sequential approach is neither intended nor recommended to be followed in practice. Rather, practitioners may run multiple steps in parallel, revisit earlier steps based on outcomes of latter steps or alter the sequence - whatever is more suited to the nature of each project. The iterative, interleaving and interactive nature of our approach, as intended, was discussed in Section 3"

Comment 16

-- Page 26, 2nd paragraph: The first phrase of that paragraph is sufficient. Figure 15 is not really explained and the rest of the paragraph starting in "Considering." is not indispensable for the discussion. I would remove them.

We reviewed both the paragraph referred to and Figure 15, to see how indispensable they are. In the end we felt that the reviewer has made a valid observation. Hence, we removed the portions referred to and merged the remnants of this paragraph with the next one, which was also rather short initially.

Comment 17 (Typos)

- Page 5, first paragraph of section iv: "itseems"
- Page 15, Section 3.5.3: do you mean "Table 6"?

- Page 18, 1st paragraph: do you mean "Table 6"?
- Page 18, 2nd paragraph: "policy.re we"
- Page 18, 2nd paragraph: do you mean "Table 6"?
- Page 19, 1st paragraph: "then"
- Page 20, 1st paragraph: do you mean "Table 7"?
- Page 21, 1st paragraph: do you mean "Table 7"?
- Page 24, 1st paragraph of Section 3.5: "We discuss some of these in Section 3.5." Do you really mean Section 3.5?

All of the above were, indeed, typos. They have now been corrected or removed.

REVIEWER #3:

Comment 1

The 6-step modeling process proposed by the authors do not seem to be general; all the steps are exemplified using "myki" system, which reduces the generality of the paper. There is no experiment section in the paper. They do not report any metrics for measuring the validity and accuracy of the simulation results. Therefore, it is hard to justify the efficiency of the proposed method. It would be much better if the authors tested the proposed approach on a real private cloud environment and report the results.

We thank the reviewer for making such a comment to help us put our work in practical perspective. The reviewer is right in saying that further experiments, beyond what is presented in the paper, would be needed to verify the conformance (or lack thereof) of simulation results with real world systems. We are aware of this and have made this point, perhaps with an unfortunately not-so-clear choice of words, in the original paper. We have now tried to clarify this in the latest version. Also, we wish to point out that this paper sought to empirically address more foundational concerns bordering on the merging of the qualitative field of goal orientation with the quantitative discipline of simulation, within the Cloud Computing domain. How do we even know that such can be done, without the exploration of existing work (which we have borrowed heavily from in this paper) and considerations for the limitations of such an effort? In addition, we wish to point out the following, in response to some of the reviewer's more specific comments, as follows:

1. **"The 6-step modeling process proposed by the authors do not seem to be general"**
   Our approach is general, although we have illustrated it using the "myki" system. We acknowledge that the original process diagram used may have given the unintended impression that the process is "myki" specific. To this end, we have replaced the process overview diagram (Figure 5) with a Gane-Sarson DFD in which all references to the "myki" have been abstracted out.
2. **"all the steps are exemplified using "myki" system, which reduces the generality of the paper"**
   Admittedly, there is a tradeoff between using examples from multiple domains (in order to show generality of applicability) and using one example from start to end (which, in our experience, is generally easier on the reader and tends to be more realistic and reproducible). Faced with the choice, we have used the latter approach, albeit with a non-trivial rather than a "toy" application. By so doing,

we think that potential readers should be able to think about how the approach might help in their own applications. Also, if multiple applications were used to illustrate, there is the danger of lengthening an already long paper, while requiring the reader to make a lot of potentially distracting mental context switches. This could be even more of a problem when it is considered that many of the essential concepts in cloud computing still lack agreed upon definitions across domains.

3. **"It would be much better if the authors tested the proposed approach on a real private cloud environment and report the results."**

   It would require a very huge research budget to pull off the creation of a private cloud testbed for the "myki" simply for experimental purposes only. We are however already working, in our lab, on experimentally verifying our approach in terms of deploying actual (relatively smaller) applications and benchmark to the cloud, based on the ideas proposed in this paper.

Comment 2

The Introduction and the Conclusion do not summarize the whole paper. The Introduction should provide an overview of the proposed approach and list the major contributions of the article.

A new paragraph has now been added to the Introduction to list some of the main contributions of the paper:

"The main contributions of this paper are as follows:

1. A method of combining goal-oriented requirements engineering techniques with simulation for cloud computing.
2. A technique for complementing the qualitative goal models in the NFR Framework [58] with a quantitative approach.
3. The development of an interactive, iterative and interleaving simulation technique based on this quantitative approach as well as stakeholder needs.
4. A new visual notation, along with heuristics and guidelines for reasoning about the goal models in the presence of the quantitative augmentation."

A new paragraph was added to the Introduction as an overview of the proposed approach:

> "At a high level of description, our approach starts with the capture stakeholder goals plus some domain characteristics such as workflows. The goals are then refined and extended quantitatively with numbers obtained from the domain, using estimation methods that we propose. All these are subsequently used to create a simulation model as a proxy for the cloud-based system architecture. Lastly, simulations are run iteratively, in an interleaving manner, against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives that have a better chance of meeting the stakeholder goals."

A summary of the approach in the paper and some of its contributions were also added as a new paragraph to the Conclusion section:

"The goal-oriented simulation approach proposed in this paper starts with the capturing and understanding of multiple stakeholders' goals, which are subsequently refined and quantitatively complemented, based on certain domain characteristics. A CloudSim simulation model is then built based on these, as the means of assessing the impact of design choices on the degree to which the various goals are satisficed. After a number of iterations, the approach yields a design which may then be tested in a real cloud deployment. Other contributions made by this paper include a new visual notation to help in managing the complexity of the modeling process as well as heuristics for reasoning about the quantitative additions to the SIG, which is a qualitative goal modeling framework."

Comment 3

Citations are parenthetical remarks; text should be readable (and grammatically correct) without them, however, this was not followed throughout the paper, for example, see the citation style in Sections 3.2, 3.5, and 4.

Every citation in the aforementioned sections, as well as in the rest of the paper has now been checked to ensure that the text is readable and grammatically correct even when the citations are removed.

Comment 4

In Section 2.3, the definitions for Cloud Service Creator and Cloud Service Provider should be reversed.

The reviewer was right. The definitions have now been reversed.

Comment 5

There are some style errors in the text, for example in citations there should be a space between the name and the reference number.

We have now corrected these earlier errors relating to the lack of space between words and the reference numbers.

Comment 6

Furthermore, Tables 1, 3, 4, 5, and 6 are in the middle of the page. They should be in either top or bottom of the page. This is also true for Figures 2, 6, 7, 8, and 11.

Having restructured the paper, all Figures and Tables in the paper are now either on the top or the bottom of the pages where they are shown.

Comment 7

Many of the figures have low quality and are hard to understand such as Figures 5, 6, 9, and 12. They are also very busy, which makes interpreting them harder.

We have made a number of adjustments to the images, including using a different format for all the images, replacing Figure 5 and making other changes to the SIGs as requested by other reviewers. Regarding the "busy" nature of the diagrams, we attribute this to the complicated relationships among the various concerns that our approach tries to capture and provide a solution to. Despite this, the images shown have been simplified as mentioned in various places in the text, for clarity and brevity. We feel that simplifying the diagrams even further may lead to a failure to capture some essential aspects that prospective adopters need to consider.

Another option we considered was to make the Figures larger. However, we opted not to do this as the paper is already quite lengthy.

Comment 8

The references are excessive.

All the references were checked individually to check how indispensable they were to the topic addressed in the paper. Where multiple references to the same topic existed, we have reduced them to the barest minimum, based on the significance of the referenced publications. Where there are multiple papers by the same author on the same subject, we have adopted only the most recent. Finally, wherever a paper could be removed, without reducing the strength of our claims, we have done so. As a result, the number of publications referenced was effectively reduced from 78 to 61 (we have added three new ones, per reviewers' suggestions).

Comment 9

There is lack of consistency in the presentation: for example in Section 3.1 there is "Figure", whereas in Section 3.3, it is "Fig.".

We have now gone through the whole paper and ensured that the short form, Fig. X, is used consistently throughout.

Comment 10

* The figures should be succeeded by their first reference, whereas in Section 3.1, Figure 5 refers to a figure in the previous page.

We have now reformatted the entire paper to ensure that all figures are either on the same page as their first reference or that preceded by their first reference.

Comment 11

* In the last paragraph of Section 3.4.1.3, the sentence ". to find out how many requests the system must be able to simultaneously" is vague.

This was a typo. The omitted word has now been added to complete the sentence.