2019

Experiential probabilistic assessment of cloud services

Mahdi Fahmidehgholami
*University of Wollongong*, mahdi@uow.edu.au

Ghassan Beydoun
*University of Technology Sydney*, beydoun@uow.edu.au

Graham C. Low
*University of New South Wales*, g.low@unsw.edu.au

Follow this and additional works at: [https://ro.uow.edu.au/eispapers1](https://ro.uow.edu.au/eispapers1)

Part of the [Engineering Commons](https://ro.uow.edu.au/eispapers1), and the [Science and Technology Studies Commons](https://ro.uow.edu.au/eispapers1)

**Recommended Citation**


Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
Experiential probabilistic assessment of cloud services

Abstract
Substantial difficulties in adopting cloud services are often encountered during upgrades of existing software systems. A reliable early stage analysis can facilitate an informed decision process of moving systems to cloud platforms. It can also mitigate risks against system quality goals. Towards this, we propose an interactive goal reasoning approach which is supported by a probabilistic layer for the precise analysis of cloud migration risks to improve the reliability of risk control. The approach is illustrated using a commercial scenario of integrating a digital document processing system to Microsoft Azure cloud platform.

Disciplines
Engineering | Science and Technology Studies

Publication Details
Experiential probabilistic assessment in adopting cloud services

Mahdi Fahmideh, Ghassan Beydoun, Graham Low
University of Technology Sydney, Australia
University of New South Wales, Australia

Abstract. Substantial difficulties in adopting cloud services are often encountered during upgrades of existing software systems. A reliable early stage analysis can facilitate an informed decision process of moving systems to cloud platforms. Such an analysis can also mitigate risks against system quality goals. Towards this, we propose an interactive goal reasoning approach which is supported by a probabilistic layer for the precise analysis of cloud migration risks to improve the reliability of risk control. The efficacy of the approach is illustrated using a commercial scenario of integrating a digital document processing system to Microsoft Azure cloud platform.

Keywords: Cloud computing adoption, legacy software systems, goal-oriented requirement engineering

1. Introduction

There are many reports of failure of cloud adoption and facing unexpected situations that have been either out of the control of service consumers or providers (Chow, Golle et al. 2009, Pepitone 2011, Linthicum 2012, Tsidulko 2016). The thought that cloud service providers are able to completely guarantee SLA may not be a realistic view in practice (Qiu, Zhou et al. 2013). Hence, one should not ignore this unavoidable fact that the uncertain risks of cloud services during their operations may impact on system quality goals leveraging them. Experiencing issues such as vendor lock-in, data security, or interoperability after moving systems to the cloud might be costly to resolve. Some examples are quite notable. For example, Infoplus’s report states that a new high-frequency cloud-based trading system started making unprofitable trades up to 40 times per second. This forced New York Stock Exchange to halt all trading and caused a loss of over $440 million (Musienko December 2017). Infoplus’ survey conducted by iLand, found that almost 57% and 44% of Amazon Web Service (AWS) and Microsoft Azure users have reported stalled or failed cloud adoption. Another survey conducted by NTT Communication concludes that although migration to the cloud is trendy, it doesn’t mean it works for all organisations. It found that 41% of the decision makers believed that migrating complex systems to the cloud is more trouble than its worth (Communications 2015).

There is a lack of complete knowledge about the actual occurrence of these risks in fact. Typically, a system architect might be asked for re-architecting of existing legacy systems to the cloud to reduce infrastructure cost and achieve higher throughput. However, the architect would be unsettled with many questions such as (i) by moving these systems to the cloud will higher system throughput be achievable in all situations? (ii) what risks are likely to occur to obstruct reducing infrastructure cost and high system throughput in the cloud? and (iii) how such risks can be negated or at least reduced in advance? A meticulous upfront analysis is required to identify probable obstacles that may cause system quality goal failures, to investigate the severity of their consequences, and to seek possible solutions beforehand. This gives them more flexibility to explore these threats and avoid premature decisions (Tran, Keung et al. 2011, C.Tang 2013, Pahl, Xiong et al. 2013). Assisting the architect to tackle such questions systematically is the focus of this paper.
In this article, we continue our research trajectory in the cloud migration but with a focus on anticipating exceptional situations that disrupt goals of cloud service adoption. In earlier work, we identified key challenges in re-architecting legacy system architectures to the cloud (Fahmideh, Daneshgar et al. 2017). Those provided both a high-level technical and a high level non-technical architectural requirements. In a more recent effort (Fahmideh and Beydoun 2018), we provided an approach to facilitate reuse of this architectural knowledge. A repository was developed to support reuse of architectural knowledge. The current paper operationalises this reuse by providing a decision making layer based on a probabilistic assessment. This enables a system architect to manage risks in adoption of cloud services. The reasoning approach is iterative and facilitates a top down refinement of obstacles. The probabilistic assessment runs concurrently with this refinement. The probability of an obstacle occurring is estimated at each level of refinement. In the decision making phase, these probabilities are then collectively used to assess their expected impact on the system quality goals. The defined precise semantic for the goal satisfaction and obstacle estimation supported by defined propagation rules enable a formal goal reasoning. This paper provides a novel probabilistic foundation for assessing obstacle severity and the concomitant degree of goal satisfaction.

The rest of the paper is organised as follows. Section 2 provides a background on legacy systems migration to the cloud following and an overview of related work. Section 3 details the framework’s components using an example of Amazon service adoption for a legacy system. Section 4 shows the applicability of the framework in a scenario of goal-obstacle analysis for moving a digital document processing systems to Microsoft Azure cloud platform. Section 5 summarises the paper and discusses limitations and future directions of the research.

2. Probabilistic goal-obstacle analysis

The goal-oriented modelling frameworks such as KAOS and i* provides means for the elicitation, elaboration, and analysis of goals from high-level strategic goals to concrete and technical details (Yu and Mylopoulos 1994). KAOS (Keep All Objects Satisfied) is a framework that supports different levels of formalism for expressing goals and reasoning about them. The formalism levels can vary from semi-formal analysis goal models to formal when a precise reasoning is required (Dardenne, Van Lamsweerde et al. 1993, Van Lamsweerde and Letier 2004). The goals are iteratively refined through top-down approach (by asking how questions to refine goals into sub-goals) as well as bottom-up approach (by asking why questions to identify parent goals). KAOS’s concepts used in this research are detailed in what follows.

Goal. A goal is a desired property or statement being expected to be satisfied by a system through the collaboration of agents or actors. Goals may vary from business level to fine-grained technical (Letier 2001). Linear temporal logic (LTL) may be used for formally representation of a goal. It is in a general form like $C \rightarrow \Theta T$ where $\Theta$ represents a LTL operator such as: o (next state), e (sometimes in the future), $\diamond$ (sometimes in the future before deadline d), $\Box$ (always in the future), $\theta \leq_d$ (always in the future up to deadline d), $W$ (always in the future unless), $U$ (always in the future until), and where $P \rightarrow Q$ means $\Box (P \rightarrow Q)$.

Obstacle. An obstacle, a technical or a non-technical one, to a goal is an exceptional situation/condition that prevents the goal from being satisfied (Potts 1995, van Lamsweerde and Letier 2000, van Lamsweerde 2004). Obstacles are the dual notion to goals. That is, goals and obstacles capture desirable and undesirable conditions, respectively (Letier 2001).

AND/OR refinement. In a goal model, goals are structured through AND/OR refinement mechanisms. They identify how goals contribute to each other. An AND-refinement link decomposes a parent goal into a set of fine-grain child goals where satisfying all child goals yield satisfying the parent goal. An AND-refinement should be complete and consistent.
A refinement is complete if all child goals suffice the satisfaction of the parent goal in a view of a specific domain. This is represented by \( \{SG_1, SG_2, SG_3, ..., SG_n, Domain\} = G \) (complete refinement). If all goals are not in contradiction, then the refinement is consistent in the domain, i.e. \( \{SG_1, SG_2, SG_3, ..., SG_n, Domain\} \neq \text{flase} \) (consistent refinement). On the other hand, OR-refinement link decomposes a goal into a set of alternative ways to satisfy a top goal and it is represented as for all \( i \): \( \{OO_i, Domain\} = O \). Goals may be in contradiction. Conflict link is can be used for this purpose but such links are out of the scope our work at this stage.

AND/OR refinements can be equally defined for obstacles. An AND-refinement means that the occurrence of the parent obstacle depends on all its child obstacles. An OR-refinement means the occurrence of a root obstacle depends on the occurrence of at least one its child sub obstacle. The completeness and consistency conditions are also the same for the obstacles. Figure 1 shows the notion is used during the goal obstacle analysis.

The proposed approach borrows the probabilistic view for goal and obstacle analysis grounded on a system-specific situation as suggested in (Cailliau and van Lamsweerde 2013). These are defined in the following.

A goal defines a possible set of behaviour. The probability of a goal satisfaction is defined in view of probability of observing such behaviours (Cailliau and van Lamsweerde 2013). For a goal \( C \rightarrow \Theta T \), the probability of satisfaction of the goal is the proportion between (i) number of possible behaviour satisfying the goal’s antecedent \( C \) and consequent \( \Theta T \) and (ii) number of possible behaviour satisfying condition \( C \). If the probability of the goal satisfaction is 1, then the goal is fully satisfied. A goal might be partially satisfied due to occurrence of some obstacles. The probability of an obstacle occurrence depends on the satisfaction of its conditions.

**Definition 1.** The probability of satisfaction of a goal in view of its possible obstructions is called estimated probability of satisfaction (EPS) (Cailliau and van Lamsweerde 2013) and is computed from the goal model. The EPS of a goal \( G \) is shown by \( P(G) \).

**Definition 2.** The minimal probability of satisfaction of a goal is called required degree of satisfaction (RDS) and is specified by existing standards and regulations in a domain of interest (Cailliau and van Lamsweerde 2013). A goal \( G \) is probabilistic if \( 0 < \text{RDS} (G) < 1 \). Note that RDS value is not unique and can vary from one scenario to another. This value may be obtained from domain experts, user experience, or the knowledge about existing systems.

**Definition 3.** Based on the EPS and RDS, the gap between estimated and expected probabilities can be measured. If \( \text{EPS} \geq \text{RDS} \), then the required goal satisfaction is reached. If \( \text{EPS} < \text{RDS} \), then the goal is not satisfied and the gap should be investigated and reduced to the extent possible. The severity of violation (SV) from a goal \( G \) is defined as:
SV(G) = RDS (G) – P(G)  

(I)

The above equations are used to measure the probability of goal satisfaction by propagating probabilities from leaf obstacles towards top goals in a goal model. Hence, the estimated probability of leaf obstacles should be provided first. These estimates are then propagated up toward root obstacles, leaf goals, and finally parent goals to identify probabilities of goal satisfaction in view of obstacle occurrence. Cailliau et. al. define the following propagation process (Cailliau and van Lamsweerde 2013):

(i) From leaf obstacles towards root obstacles. The system architect should rely on domain information to estimate the probability occurrence of leaf obstacles in the refinement goal model. The information sources include the specification of cloud services, system developers or end-users’ experience, consultation with domain experts, and statistical data about legacy systems which can be obtained through techniques like interview or Delphi method.

In an AND-refinement, a parent obstacle occurs if all its sub-obstacles (SO) also occur. Thus, the probability of the parent obstacle equals the probability of all sub-obstacles and their combined occurrence towards the satisfaction of the parent obstacle. This is computed as follows:

\[ P(O) = P(SO_1) \times P(SO_2) \times P(SO_3) \times \ldots \times P(O|SO_1, SO_2, SO_3, \ldots) \]  

(II)

In equation (I), the architect also needs to know from the domain information how often the occurrence of leaf obstacles O1, O2, O3, and … causes the parent obstacle O happens.

For an OR-refinement, the probability of the parent obstacle not occurring which equals the probability that none of the children obstacles yield a satisfaction of the parent obstacle. This is computed as follows:

\[ P(O) = 1 - (1 - P(SO_1) \times (1 - P(SO_2) \times (1 - P(SO_3) \times \ldots \ldots)) \]  

(III)

(ii) From root obstacles towards leaf goals. The probability of not satisfying of a leaf goal (LG) is given by the probability that the root obstacle occurs (RO) and such occurrence results in not satisfying of the leaf goal. This is presented using the following equation:

\[ 1 - P(LG) = P(RO) \times P(\neg LG|RO) \]  

(IV)

If the leaf goal is obstructed by multiple obstacles, then the goals is satisfied when any obstacles occurs. This is computed as follows:

\[ P(LG) = (1 - P(O_1) \times P(\neg LG|O_1)) \times (1 - P(O_2) \times P(\neg LG|O_2)) \ldots \]  

(V)

(iii) From leaf goals towards root goals. The satisfaction probability of a parent goal depends on probabilities of satisfaction of its leaf goals. Thus, the reduced degree of satisfaction of an obstructed leaf goal should be propagated upwards in the goal model in order to specify consequences of all obstacles. For example, a parent goal with two leaf goals is satisfied if all the leaf goals are satisfied, or satisfaction of the first goal is sufficient to satisfy the parent goal, or the satisfaction of the second one is sufficient to satisfy the first one (Cailliau and van Lamsweerde 2013). To specify the consequence of the obstacles to the quality goals, the computed probabilities for all leaf obstacles are propagated up towards goals to know how much resulting estimated probability of satisfaction (EPS) of higher-level goals in view of its possible obstacles deviates from their required degree of satisfaction (RDS). A parent goal is satisfied if its sub goals are satisfied. This leads to the following equation:

\[ P(G) = P(SG_1, SG_2) \times P(G|SG_1, SG_2) + P(SG_1, \neg SG_2) \times P(G|SG_1, \neg SG_2) + P(\neg SG_1, SG_2) \times P(G|\neg SG_1, SG_2) + P(\neg SG_1, \neg SG_2) \times P(G|\neg SG_1, \neg SG_2) \]  

(VI)
3. Proposed approach

The approach has a four-step process starting with elicitation of high-level goals for the cloud adoption to empower legacy systems and ending with a set of critical obstacles obstructing goal satisfactions. The output artefact of the approach is a list of critical obstacles that are required to be dealt. For the explanatory purpose, we use an example scenario of moving the data storage of a legacy system to Amazon Simple Storage Service (S3) so that end users can get the content directly from Amazon S3. S3 provides a secure, durable, highly-scalable cloud storage to store and retrieve any amount of data from anywhere on the web (AmazonS3).

As alluded earlier, the knowledge repository has collections of goals, obstacles, and resolution tactics which will be used to illustrate the goal analysis. The repository itself was developed in (Fahmideh and Beydoun 2018). The steps of the approach are detailed in what follows.

Step 1. Specifying goals

The system architect should identify goals that are expected to be satisfied by integrating the system with cloud services. In the current scenario, deploying the legacy system database on Amazon S3 are expected to positively contributes towards five root goals namely Achieve [Reduced IT cost], Achieve [Improved response time], Achieve [Improved availability], and Achieve [Improved consistency]. For example, the root goal Achieve [Improved response time] has the following specification.

Goal Achieve [Improved response time]
Category Performance Goal
Definition [Using Amazon S3, transferring 100 terabyte live data stream with internet connection 1000 Mbps at 80% network utilization from local network should not take no more than one week].
Formal spec ∀ d: data, (d.submitted → ◊ ≤7 days d.processed )
RDS 95%.

This goal is satisfied if its two child goals are satisfied. In other words, AND-refinement for the parent goal Achieve [Improved response time] captures a combination of two leaf goals Achieve [Reduced data uploading time] and Achieve [Reduced query processing time] entailing the parent goal should be completely satisfied (Figure 2). The specification of the goal Achieve [Reduced data uploading time] is:

Goal Achieve [Reduced data uploading time]
Category Performance Goal
Definition [Using Amazon S3, transferring 1 terabyte live data with internet connection 1000 Mbps at 80% network utilization from local network should not take no more than 24 hours].
Formal spec ∀ d: data, (d.submitted → ◊ ≤24 hours d.processed )
RDS 90%.

Figure 2 Refining the parent goal Achieve [Improved response time] to child goals
**Step 2. Identifying obstacles**

Goals that are positively contributed by using cloud services are stated without a realistic view of potential obstacles that might occur in an operational environment. For each quality goal, the system architect explores potential obstacles and shortlists ones that are probable to occur against quality goals (Figure 3). Identification of obstacles is based on information sources such as developers, user experience, statistical data, and available technical accounts about Amazon S3. In this scenario, former developers’ experience concedes that the leaf goal Achieve [Reduced data uploading time] is likely hampered by two obstacles *Performance variability of Amazon S3* and *Geographical distance*.

In addition, for the purpose of message processing, the current legacy system uses Kafka technology, i.e. a common open source technology for massive scale publishing and subscribing message queues. Integrating this technology with Amazon S3 storage may cause latency for the data consistency of the system. According to the past experience of the developers in using Amazon S3, there is no guarantee of the exact time to upload the system data to the S3 servers due to their workload unpredictability. The uploaded data by a user may be stored in S3 data storage for an extended period of time which may cause a data inconsistency issue. This obstacle, named here as *Latency for moving data from Kafka to S3*, is against the goal Achieve [Improved consistency]. As shown in Figure 3, other leaf obstacles that are modelled by the system architect are *Service transient fault*, *S3 outage*, *Department downsizing*, and *Extra management effort per annum*. The obstacles *Extra cost of training new data integrator* and *Extra cost for monitoring tools* are refinements of obstacle *Extra management effort per annum*. The obstacles *High uploading time for blob* and *Low throughput to write bucket* are refinements of *Performance variability of Amazon S3*. *Local electrical storm*, *S3 power outage*, *S3 data centre outage*, *I/O issues of servers*, and *Local network disruption* are refinements of the obstacle *Service outage*. Figure 4 shows all refinements to root obstacles.

No obstacle is identified in this scenario for the goal Achieve [Reduced query processing time]. AND/OR refinements are used to show sub obstacles of a root obstacle. AND-refinements include a combination of sub obstacles that aggregately cause the occurrence of a parent obstacle. OR-refinements represent a set of alternative sub obstacles where the occurrence of each will cause the parent obstacle.

![Figure 3. Identified obstacles against achieving quality goals Achieve [Reduced IT cost], Achieve [Reduced response time], Achieve [Improved consistency], and Achieve [Improved availability]](image-url)
**Step 3. Assessing probability of obstacles**

Table 1 shows a sample of 12 leaf obstacle estimations based on the developers’ opinions and capturing statistics about Amazon S3 server, memory and I/O usage, and legacy system performance. These estimates can be refined as the goal analysis proceeds.

<table>
<thead>
<tr>
<th>Obstacle</th>
<th>Definition</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department downsizing</td>
<td>Moving legacy system database to S3 definitely cause change in roles and responsibilities defined in the maintenance team of IT department.</td>
<td>1</td>
</tr>
<tr>
<td>Extra cost of training new data integrator</td>
<td>Although developers have expertise in using legacy-based tools for data integration, they require learning new data integrator tools that are specific for combining and import/export legacy data and S3.</td>
<td>0.6</td>
</tr>
<tr>
<td>Extra cost for monitoring tools</td>
<td>New tools should be installed and new role should be appointed for monitoring the database performance in S3.</td>
<td>0.5</td>
</tr>
<tr>
<td>Geographical distance</td>
<td>It is likely that buckets are stored in S3 servers are located in distance far from the local network of the company which may cause high uploading time.</td>
<td>0.04</td>
</tr>
<tr>
<td>High uploading time for blobs (100k entries)</td>
<td>Over one week measurements of storing ten set of 100k buckets in S3, it was observed that transferring one of the buckets took more than 24 hours.</td>
<td>0.2</td>
</tr>
<tr>
<td>Low throughput to write buckets</td>
<td>In high workload, one set of 100k bucket were stored in the database with high delay.</td>
<td>0.1</td>
</tr>
<tr>
<td>Latency for moving data from Kafka to S3</td>
<td>In almost 60 percentages of cases, there is high latency for offline moving data from Kafka to S3 storage. Due to server workload, it is highly probable that end users may observe data that are stale, not from the most recent update.</td>
<td>0.6</td>
</tr>
<tr>
<td>Local electrical storm</td>
<td>The company is located in rainy geographical area and has experienced electrical storm damaged power equipment at one of the data centres.</td>
<td>0.01</td>
</tr>
<tr>
<td>S3 data centre power outage</td>
<td>Drop time of Amazon S3 servers is unlikely to occur.</td>
<td>0.001</td>
</tr>
</tbody>
</table>
Local network disruption
Network misconfigurations or the level of workload may cause a local network disruption. 0.02

I/O issues of servers
I/O issues may occur with very low probability but still probable. 0.001

Transient fault of service
Developers’ experience shows the probability of transient faults when legacy system tries to connect cloud services. 0.01

The probabilities of the occurrence of the obstacles in the goal model are now computed from leaf obstacles towards high-level goals. Depending on the structure of a goal model, assessing the probability of obstacles and their consequences has three possible steps (Cailliau and van Lamsweerde 2013):

(i) From leaf obstacles towards root obstacles. As noted in Section 2, in an AND-refinement, a parent obstacle occurs if all its leaf obstacles occur. Similarly, in OR-refinement, a parent obstacle occurs in the case of occurrence of any leaf obstacles. Given the probabilities of leaf obstacles in Table 2, the propagation rules for AND/OR-refinements in equations (II) and (III) are applied bottom-up to compute probabilities for the parent obstacles Extra management effort per annum, performance variability of Amazon S3, and S3 data centre outage.

With respect to the obstacle performance variability of Amazon S3, the estimations for the leaf obstacles are namely 20% of data transfer from local network to Amazon S3 experiences a high uploading time, 4% uploading data are performed on servers which are too far from the local network. From domain information, the system architect knows that 95% a high data uploading time with low throughput to write buckets is due to the performance variability of Amazon S3. Given that, the propagation rule for the AND-refinement results in the probability of the root obstacle Performance variability of Amazon S3:

\[
P(\text{performance variability of Amazon S3}) = P(\text{High uploading time for blogs}) \times P(\text{Low throughput to write buckets}) \times (P(\text{performance variability of Amazon S3 | High uploading time for blogs, Low throughput to write buckets}) = 0.2 \times 0.1 \times 0.95 = 0.019
\]

This means that the performance variability of S3 for the data processing occurs in almost 2% of cases.

The propagation rule for OR-refinement is used for the root obstacle Extra management effort per annum:

\[
P(\text{Extra management effort per annum}) = 1 - P(\text{Extra cost of training new data integrator}) \times P(\text{Extra cost for monitoring tools}) = 1 - (1 - 0.5 \times 0.99) \times (1 - 0.6 \times 0.99) = 0.205
\]

In above the proportion of both obstacles is considered 99%. Similarly, the probability of S3 data centre power outage is computed using the equation III as follows:

\[
P(\text{S3 data centre outage}) = 1 - (1 - P(\text{Local electrical storm}) \times P(\text{S3 data centre power outage | Local electrical storm})) \times (1 - P(\text{S3 power outage}) \times P(\text{S3 data centre power outage | S3 power outage})) = 1 - (1 - 0.01 \times 0.99) \times (1 - 0.001 \times 0.98) = 0.010
\]

According to the statistical data, there is a 2% probability of local network disruption, 0.01% of I/O issues of servers, 0.1% S3 data centre power outage, 1% Local electrical storm, and 1% S3 power outage. The proportion of local network disruption, I/O issues of servers, and S3 data centre power outage is respectively 99%, 98%, and 100%. The propagation rule for OR-refinement in equation (III) yields the following probability for the parent obstacle S3 outage:

\[
P(\text{S3 outage}) = 1 - (1 - 0.02 \times 0.99) \times (1 - 0.001 \times 0.98) \times (1 - 0.01 \times 1) = 0.03
\]

The above value means the S3 outage occurs in 3% of cases.
(ii) From root obstacles towards leaf goals. Back to Figure 4, the goal Achieve [Reduced IT cost] is satisfied when none of the leaf obstacle occurs. The probability of satisfaction for this goal is computed using equation V which is:

\[ P(\text{Achieve [Reduced IT cost]}) = (1 - P(\text{Extra management effort per annum}) \times P(\neg \text{Reduced IT cost} | \text{Extra management effort per annum}) \times (1 - P(\text{Department downsizing}) \times P(\neg \text{Reduced IT cost} | \text{Department downsizing})) = (1 - 0.6 \times 0.6) \times (1 - 1 \times 1) = 0 \]

This means that using Amazon S3 will not certainly reduce the IT cost unlike the initial expectation.

In addition, the probability of satisfaction for the leaf goal Achieve [Reduced data uploading time] is computed using equation IV which is:

\[ P(\text{Achieve [Reduced data uploading time]}) = (1 - P(\text{Performance variability of Amazon S3}) \times P(\neg \text{Reduced data uploading time} | \text{Performance variability of Amazon S3}) \times (1 - P(\text{Geographical distance}) \times P(\neg \text{Reduced data uploading time} | \text{Geographical distance})) = (1 - 0.02 \times 0.02) \times (1 - 0.04 \times 0.04) = 0.9 \]

The above value means in 90% of cases, 1 terabyte live data with internet connection 1000 Mbps at 80% network utilization, will be transferred from the local network to S3 within the prescribed 24 hours. Furthermore, using the same equation, the probability of the goal Achieve [Availability] is:

\[ P(\text{Achieve [Availability]}) = (1 - P(\text{S3 outage}) \times P(\neg \text{Availability} | \text{S3 outage}) \times (1 - P(\text{Service transient fault}) \times P(\neg \text{Availability} | \text{Service transient fault})) = (1 - 0.03 \times 0.03) \times (1 - 0.01 \times 0.01) = 0.9 \]

(iii) From leaf goals towards root goals. In the exemplar model (Figure 4), the system architect checks if the model satisfies the expected threshold which is the response time for processing of requests, i.e. goal Achieve [Improved runtime response], and this should be satisfied in 90% of cases (RDS=0.9). For the leaf goal Achieve [Reduced data uploading time] the computed satisfaction probability is 0.9. Based on the developers’ experience, there is no obstacle against the goal Achieve [Reduced query processing time]. Thus, the probability of satisfaction of the goal Achieve [Improved response time], which is an AND-refinement link, is 1 \times 0.9 = 0.9. The resulting EPS for this goal is 90% which means the adopting Amazon S3, as modelled, is not able to satisfy the expected standard 95%.

**Step 4. Identifying critical obstacles**

The system architect may be interested in the identification of leaf obstacles that cause a severe violation. For this purpose, one single leaf obstacle is considered and the probabilities for the rest of leaf obstacles are set to 0. Now, propagation from leaf obstacles towards the root goal is performed and a violation severity for the root goal is computed.

It should be noted that some leaf obstacles may have small probabilities but they are more important than others (Cailliau and van Lamsweerde 2013). Furthermore, it is important to realise that in the case of generating many leaf obstacles, the computation of obstacle consequences on leaf goals and root goals might be difficult. In fact, the identification of critical obstacles is a multi-criteria optimisation problem where the aim is to find the minimal set of leaf obstacles that maximise the violation severity of high-priority goals. It includes three steps: (i) generating all leaf obstacle combinations, (ii) computing SV(G) for each obstructed goal (if required, goals are weighted), and (iii) sorting leaf obstacles combination based on their severity. Identification and prioritisation of critical obstacles, which is not the focus of our approach, can be performed using common techniques covered elsewhere such as AHP and
brute-force technique. The critical obstacles should be tackled prior to the enactment of cloud services. The obstacle handling is the focus of (Fahmideh and Beydoun 2018) where a catalogue of resolution tactics are classified and stored in the approach’s to be by a system architect to deal with the critical obstacles.

4. Validation

The case study used for the validation is based on the scenario of moving a Web-based Digital Document Processing (DDP) legacy system to the cloud (Rabetski 2012, Rabetski and Schneider 2013). InformIT is a small independent software vendor in Sweden that has developed the DDP to offer services to medium and large organisations with adequate infrastructure and technicians. InformIT planned to expand the system’s services around those small companies that could not afford the financial commitment to use the system such as being charged per user and installation. Deploying the DDP in the cloud would enable small companies to utilise its services without purchasing for infrastructure. The system architect was interested in analysing obstacles that such a transition would experience and accordingly handle them beforehand. The goal modelling steps were performed as described in the following.

Step 1.Specifying goals. The system architect identified four goals Achieve [Performance], Achieve [Integrity], Achieve [Portability], and Achieve [Accountability] to be satisfied by the moving DDP to Microsoft Azure cloud platform. These goals are defined as follows:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Category</th>
<th>Definition</th>
<th>Formal spec</th>
<th>RDS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Security goal</td>
<td>DDP’s documents should not be accessible/readable by other tenants that are running on same Azure servers.</td>
<td>$\forall \text{doc: DDP’s document} \rightarrow \Box \text{doc.processed by registered tenant}$</td>
<td>100%</td>
</tr>
<tr>
<td>Performance</td>
<td>Performance goal</td>
<td>acceptable system throughput for rendering a digital document should be no more than 4.9 seconds.</td>
<td>$\forall \text{doc: document, doc.submitted} \rightarrow \Diamond \leq 4.9\text{ seconds doc.processed}$</td>
<td>95%</td>
</tr>
<tr>
<td>Testability</td>
<td>Testing goal</td>
<td>the whole process of performing various tests scripts of system components should be doable within a specific time limit.</td>
<td>$\forall \text{com: DDP's component} \rightarrow \Diamond \leq 1\text{ working day com.tested}$</td>
<td>95%</td>
</tr>
<tr>
<td>Integrity</td>
<td>Integrity goal</td>
<td>system components should be able to invoke cloud services.</td>
<td>$\forall \text{com: DDP's component} \rightarrow \Box \text{com.invoked, cloud services}$</td>
<td>95%</td>
</tr>
<tr>
<td>Portability</td>
<td>Portability goal</td>
<td>DDP’s documents should be readable and processable in both platforms.</td>
<td>$\forall \text{doc: DDP's document} \rightarrow \Box \text{doc.processed}$</td>
<td>95%</td>
</tr>
</tbody>
</table>
Step 2. Identifying obstacles. For each goal, the system architect refined the top goals towards root obstacles and subsequently leaf ones that may cause dissatisfying quality goals. Information provided by developers and end users of DDP were the main source used to check if an obstacle was likely to occur in this scenario. For example, consider the goal Achieve [Integrity]. The current DDP’s APIs were not be compatible with their counterparts in the Microsoft Azure cloud platform. Thus, the leaf goal is obstructed by the root obstacles Incompatible APIs (i.e. Legacy’s APIs and Microsoft Azure) and Incompatibility of legacy data storage and cloud. Furthermore, the parent obstacle Incompatibility of legacy data storage and cloud was also refined into two leaf obstacles which were both domain specific instantiation of the obstacle Incompatibility of legacy data storage and cloud. The definition of the leaf obstacles against the goal Achieve [Integrity] depicted in Figure 5 and defined as follows:

**Obstacle Incompatible APIs**  
**Definition** [DDP uses API’s offered by .NET 2.0 and Visual Studio 2005 which may not be compatible with Microsoft Azure platforms].  
**Formal spec** $\diamond (¬\text{DDP APIs Incompatibility})$

**Obstacle Incompatible datatypes**  
**Definition** [DDP datatypes are based on SQL Server Database .NET 2.0 platform which might not be compatible with Microsoft Azure database solution].  
**Formal spec** $\diamond (¬\text{DDP Datastorage Compatibility})$

**Obstacle Incompatible data operations**  
**Definition** [The data operations supported by SQL Server Database .NET 2.0 platform might not be compatible with Microsoft Azure database solution].  
**Formal spec** $\diamond (¬\text{DDP DataOperation Compatibility})$

Confirmed by the past experience of developers, the goal Achieve [Performance] could be possibly obstructed by the performance variability of Microsoft Azure servers which could be out of the control of developers. This was specified by the obstacle Microsoft Azure Middleware latency in the goal model. This parent obstacle was also refined into three leaf
obstacles *Microsoft Azure database middleware latency*, *Microsoft Azure message middleware latency*, and *Microsoft Azure transaction middleware latency*. Other relevant and potential obstacles against the goal *Achieve [Performance]* were *Distance from Microsoft Azure servers*, *Microsoft Azure transaction middleware latency*, and *On-premise hardware latency* (Figure 6). The step also resulted in refinements of some other root obstacles as shown in Figure 7.

Figure 6 leaf obstacles against goal *Achieve [Performance]*

In total eighteen leaf obstacles that were identified in this step (Figure 5, 6, and 7) that were annotated with the estimation of their probabilities. Table 2 shows the leaf obstacle estimates. The estimates have been based on personal architect’s opinion with consultation with developers, and observations from the DDP performance. These estimates can be later refined once real statistical data are available.

Figure 7 leaf obstacles against goals *Achieve [Security]*, *Achieve [Testability]*, *Achieve [Portability]*
Table 2 estimated probabilities for the leaf obstacles

<table>
<thead>
<tr>
<th>Leaf obstacle</th>
<th>Probability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch between DDP and Azure Storage API file systems</td>
<td>1</td>
</tr>
<tr>
<td>Incompatible APIs</td>
<td>0.72</td>
</tr>
<tr>
<td>Incompatible datatypes</td>
<td>0.5</td>
</tr>
<tr>
<td>Incompatible data operations</td>
<td>0.3</td>
</tr>
<tr>
<td>Performance variability of Microsoft Azure servers</td>
<td>0.07</td>
</tr>
<tr>
<td>High-time for session handling</td>
<td>0.01</td>
</tr>
<tr>
<td>Microsoft Azure message middleware latency</td>
<td>0.09</td>
</tr>
<tr>
<td>Microsoft Azure database middleware latency</td>
<td>0.09</td>
</tr>
<tr>
<td>Microsoft Azure transaction middleware latency</td>
<td>0.08</td>
</tr>
<tr>
<td>Distance from Microsoft Azure servers</td>
<td>0.05</td>
</tr>
<tr>
<td>Browser latency</td>
<td>0.05</td>
</tr>
<tr>
<td>On-premise DDP hardware latency</td>
<td>0.09</td>
</tr>
<tr>
<td>Session hijacking</td>
<td>0.03</td>
</tr>
<tr>
<td>Code alteration</td>
<td>0.01</td>
</tr>
<tr>
<td>Code control</td>
<td>0.02</td>
</tr>
<tr>
<td>Insecure data location</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Step 3. Assessing Obstacles. The various probabilities were used to compute the impact of leaf obstacles on top goals via the propagation equations as described in the following.

To compute the probability of obstacle Incompatibility of DDP data storage and cloud, the probability of its two child obstacles should be calculated first. Given the probability estimation for the leaf obstacles Incompatible datatypes and Incompatible data operations from Table 2 and considering the proportion for having incompatible datatypes or data operations causing incompatibilities between DDP and S3 is, respectively, 0.99% and 98%, the probability of the parent obstacle Incompatibility of DDP data storage and cloud is obtained using the equation III:

\[
P(\text{Incompatibility of DDP data storage and cloud}) = 1 - (1 - 0.5\times0.99) \times (1 - 0.3\times0.98) = 0.65
\]

The probability of the obstacle Incompatible APIs is 0.72. The proportion Incompatible APIs and Incompatibility of DDP data storage and cloud are 0.99% and 98%. Thus, the probability of occurrence for the parent obstacle Incompatibility of DDP and cloud service can be again computed using the equation III which is:

\[
P(\text{Incompatibility of DDP and cloud service}) = 1 - (1-0.65\times0.99)(1-0.72\times0.98) = 0.89
\]

The above probability is then propagated towards the corresponding top goal and thus the resulting EPS for the top goal Achieve [Integrity] is:

\[
\text{EPS (Achieve [Integrity])} = 1 - 0.89 = 0.11
\]

RDS for the goal Achieve [Integrity] is 1 meaning that DDP’s components should be fully (100%) integratable with cloud services without occurring incompatibilities. Using equation I, SV for the goal Achieve [Integrity] is 0.95 – 0.11 = 0.84. This means the leaf obstacles are critical and should be resolved.

Similar computations are performed for the goal Achieve [Performance]. Firstly, through the OR-refinement, the root obstacle Microsoft Azure Middleware latency is refined into three domain-specific sub-obstacles namely Microsoft Azure database middleware latency, Microsoft Azure message middleware latency, and Microsoft Azure transaction middleware latency. Using propagation equation III, the probability of the root obstacle Microsoft Azure Middleware latency is:

\[
P(\text{Microsoft Azure Middleware latency}) = 1 - (1-0.01\times0.99)(1-0.09\times0.98)(1-0.08\times0.99) = 0.17
\]
The obstacle Service latency is refined into Distance from Microsoft Azure servers, On-premise hardware latency, and Browser latency. Again, using equation II, the computed root obstacle Network latency is:

\[ P(\text{Service latency}) = 1 - (1 - 0.05*0.99)*(1 - 0.09*0.99)*(1-0.05*0.99) = 0.17 \]

Using equation V, the probability of goal satisfaction for Achieve [Performance] is:

\[ \text{EPS (Achieve [Performance])} = 1 - (1-0.07*0.99)*(1-0.01*0.99)*(1-0.17*0.99)*(1-0.17*0.99) = 0.36 \]

RDS for the goal Achieve [Performance] is 0.95. But SV for the goal Achieve [Performance] = 0.95 – 0.36 = is 0.6 meaning that moving the DDP to Microsoft Azure cloud platform does not render digital documents within time limit 4.9 seconds provide in 60% of cases. Countermeasure should be taken into account to satisfy the goal Achieve [Performance].

Furthermore, the satisfaction of the goals Achieve [Testability] and Achieve [Portability] depend, respectively, on the obstacles Microsoft Azure Middleware latency and Switch between regular file system API to Microsoft Azure Storage API. Thus:

\[ \text{EPS (Achieve [Testability])} = 1 - 0.17 = 0.83 \]
\[ \text{EPS (Achieve [Portability])} = 1 - 1 = 0 \]

These values are far from RDS 1 prescribed on these goals. The system architect thus should carefully investigate these critical obstacles.

The probability of satisfaction of the goal Achieve [Security] depends on the impact of the probability of leaf obstacles on the leaf goals. Given the probability estimates for the leaf obstacles in Table 2 and shown in the model fragment in Figure 5, the system architect propagate probabilities according to the rules in Section 2 to compute the probability of the parent obstacle Code disruption.

\[ P(\text{Code disruption}) = 1 - (1-0.01*0.99)*(1-0.02*0.99) = 0.02 \]

The architect may then compute the probabilities for the root obstacle Data disclosure:

\[ P(\text{Data disclosure}) = 1 - (1-0.02*0.99)*(1-0.03*0.99) = 0.04 \]

The above probability may be then propagated to the corresponding obstructed leaf goal Achieve [Data confidentiality]:

\[ P(\text{Achieve [Data confidentiality]}) = 1 - 0.04 = 0.96 \]

On the other hand, the probability of the goal Achieve [Data location security] is:

\[ \text{EPS (Data location security)} = 1 - P(\text{Insecure data location}) = 1 - 0.001 = 0.99 \]

Now, the results are propagated from leaf goals Achieve [Data confidentiality] and Achieve [Data location security] towards the root goal Achieve [Security] to determine all obstacle consequences. In other words, for the goal Achieve [Security], the propagation rule resulted in the following goal satisfaction probability using equation VI.

\[ \text{EPS (Achieve [Security])} = 0.96*0.98 = 0.94 \]

This means that the probability of satisfying DDP security when running on Microsoft Azure cloud platform is about 0.94 if all leaf obstacles are correctly estimated. This is less than expected RDS prescribed in the definition of the goal Achieve [Security]. The partial model in Figure 8 shows the overall satisfaction of the goals.
Step 4. Identifying critical obstacles. Eleven leaf obstacles were found to be severe against the goals as shown in Table 3. For example, *Switch between DDP and Azure Storage API file systems* with estimated probability 1, causes a violation severity of .95 for the root goal Achieve [Portability]. This indicates that incompatibility between DDP and cloud platforms is inevitable. The leaf obstacle *Microsoft Azure database middleware latency* with the estimated probability of 0.09 causes a violation severity 0.04 for the root goal Achieve [Performance]. Therefore, in the next step of risk management the architect should define mechanisms in the new cloud-based architecture to reduce the gaps between RDS and EPS and lower them as much as possible.

As mentioned earlier, some leaf obstacles may have small probabilities whilst they are actually more critical than others. For instance, the leaf obstacle *Code alteration* with a probability of 0.01 may obstruct the leaf goal Achieve [Data confidentiality] and subsequently the parent goal Achieve [Security]. Although the estimated probability is low, still the obstacle occurrence might be critical and negatively affect the whole security of DDP.

<table>
<thead>
<tr>
<th>Table 3. Critical obstacles against goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaf obstacle</td>
</tr>
<tr>
<td>Incompatible APIs</td>
</tr>
<tr>
<td>Performance variability of Microsoft Azure servers</td>
</tr>
<tr>
<td>Microsoft Azure message middleware latency</td>
</tr>
<tr>
<td>Microsoft Azure database middleware latency</td>
</tr>
<tr>
<td>Microsoft Azure transaction middleware latency</td>
</tr>
<tr>
<td>On-premise DDP hardware latency</td>
</tr>
<tr>
<td>Session hijacking</td>
</tr>
<tr>
<td>Code alteration</td>
</tr>
<tr>
<td>Code control</td>
</tr>
<tr>
<td>Insecure data location</td>
</tr>
<tr>
<td>Switch between DDP and Azure Storage API file systems</td>
</tr>
</tbody>
</table>

5. Related work

Our research falls into the literature area of requirements and risk analysis of cloud services. There are numerous works have looked at this problem. In what follows, we review some of this literature, from least to most related ones to ours.

The less related works deal with risk analysis at a very high-level. They provide insights to system architects on system integration issues with cloud services. However, they do not provide a solution to examine risk likelihood and severity during a cloud architecture design phase. For example, Heiser et. al. reports an analysis of unique attributes of obstacles related to the security goal such as data integrity and segregation through which the authors found
that factors such as the location independence of service provider subcontracting may endanger system security (Heiser and Nicolett 2008).

On the other hand, more closely related works offer solutions for dealing with migration risks. For instance, plenty of studies focus on security risk analysis e.g. Chen et. al. developed a framework for the automatic detection of conflicts and inconsistencies in user requirements and organisational policies (Chen, Yan et al. 2012). The framework suggests how cloud services may satisfy these requirements and policies. Their prototype shows how managing heterogeneous cloud infrastructure services could be undertaken for large organisations.

Martens et. al., developed a quantitative model to balance costs and risk factors for outsourcing decisions regarding cloud service adoption (Martens and Teuteberg 2012). Mouratidis et. al. incorporated a modelling language along with a structured process to identify security and privacy requirements to select cloud providers based on the satisfiability of the service provider to the relevant security and privacy requirements (Mouratidis, Islam et al. 2013). Shirvani et. al., present a framework through which a module is added to a cloud server broker to log security information and then quantify the cloud security risks (Hosseini Shirvani, Rahmani et al. 2018). All these studies can be complemented by our work as they can use our formalism to represent goals and AND/OR refinement mechanisms to get more in-depth analysis of security risk. A clear advantage of our approach is its wide applicability to assess other risk management goals.

Perhaps, the closest work to our research is Saripalli et. al. in which they take into account probabilities of risks (Saripalli and Walters 2010). This work presents a probabilistic approach, called QUIRC (Quantitative Impact and Risk Assessment), for analysing and assessing typical security attacks to systems deployed in the cloud. To assess the impact of risks on system security, a combination of the probability of security threats and their severity are computed. The approach also benefits from the available data from SANS (System Administration, Networking, and Security Institute) to facilitate estimations of security risks. Compared to QUIRC, one key advantage of our approach is that it allows an in-depth refinements of obstacles. The impact of the lower level details is then collectively used to assess risk. This further enables architects to contrast and to better understand the goal satisfaction in the view of any obstacles. Unlike QUIRC, our approach is interactive and participatory using domain information to estimate risk probabilities instead of merely using public data as the main source for estimations.

Goal-oriented approach has also been accommodated in other works. For instance Islam et al. analyse security and privacy risks of cloud services (Islam, Mouratidis et al. 2013) using goals and risks identified from sources such as data, service/application, technical, and organisational measures. A weakness of this work is that the entire risk analysis is treated as one single security goal without subsequent refinement. Zardari et. al. also use a goal-oriented approach to model obstacles in adopting cloud services and then employs Analytical Hierarchy Process (AHP) to prioritise obstacles based on their weights (Zardari, Bahsoon et al. 2014). Step 4 of our approach can be viewed complementary to perform obstacle prioritization.

Unlike our earlier work presented in (Fahmideh and Beydoun 2018), which highlights the need for an obstacle analysis and reusing empirical knowledge of architecture design, the proposed approach in this research is unique in that it provides a probabilistic foundation for measuring the satisfaction of arbitrary goals. It takes into account a model refinement through with finer granularity. The refined goals and obstacles are easy to measure and provide an operational assessment of high-level goals.
6. Summary, limitation, and future work

In this paper, we have presented an approach for analysing risks in integrating legacy systems with cloud services. The approach which is based on goal reasoning defines an identify-assess-resolve loop and leverages probability foundation to formalise goals for adopting cloud services in terms of their satisfaction and obstacles causing them fail. System architects can benefit from the approach to explore obstacles at the early stage of cloud migration when there is more flexibility. The approach has been applied in scenarios of deploying a legacy system database and a digital document processing system to Amazon S3 and Microsoft Azure cloud platform.

A limitation of the approach is its reliance on domain information and personal judgment for estimating the probability of occurrence of obstacles which is used for computing goal satisfaction and violation, i.e. EPS and SV. The accuracy of estimates for leaf obstacles are crucial to getting reliable estimation for EPS and SV of top goals. Identifying estimations may impose excessive overhead to the decision making process in the case of a model with a large number of obstacles. The getting continuous feedback from system developers and end users for the iterative model refinement are important means to alleviate the issue of subjective decisions.

While we believe that our approach is generic in nature and provides system architects basic modelling techniques and goal/obstacle estimation, we certainly do not claim that the approach validation is conclusive. The approach is yet to be used in a complete migration project. There are still a few validation steps required to compare the risk estimates and actual observed risks in a project. This will produce more accurate assessment of the produced results. We plan to further validate and compare estimated risks at the early stage of the migration project, and the collected risk information at the post migration stage. This will give further insights on how accurate estimates produced by our approach are. This will enable us to appraise the reliability of the approach. Towards this, we just developed a prototype system to model goals and obstacles (CCER 2018).

We intend to extend our approach in two pertinent directions. Due to the interactive nature of the framework, the automating reasoning and maintenance of goal models is necessary especially when the scale of goal modelling is increased. We are currently working on developing a tool suite that automates and the computations of probabilities in a consistent manner. Finally, we are also developing capability of considering the interdependencies among goal and obstacles in model elements.

Reference

AmazonS3 "Amazon Web Services S3 - Simple Cloud Storage Service." https://aws.amazon.com/s3/?sc_channel=PS&sc_campaign=acquisition_AU&sc_publisher=google&sc_medium=s3&sc_content=s3_e&sc_detail=s3.amazonaws.com&sc_category=s3&sc_segment=118649900484&sc_matchtype=e&sc_country=AU&sc_detail=s3.amazonaws.com&ef_id=Uvy8OgAABEYZb5XA:20161022040726:s (Last accessed October 2016).


Rabetski, P. (2012). "Migration of an on-premise application to the cloud."


Zardari, S., et al. (2014). "Cloud Adoption: Prioritizing Obstacles and Obstacles Resolution Tactics Using AHP."