COMPOSING CONTEXT-AWARE DISTRIBUTED SYSTEMS USING QoS AND TRUST PRINCIPLES

Dimuthu U. Gamage, Lahiru S. Gallage, Rajeev R. Raje
Dept. of Computer and Information Science
Indiana University-Purdue University Indianapolis, Indianapolis, IN, USA
Email: {dcundupi, lspileth, rraje}@cs.iupui.edu

Abstract

The emergence of ubiquitous computing and the wide adoption of smart phones over the past few years require many Web Services to function in a context-aware manner. In such services, not only the functional attributes, but also the QoS attributes (e.g., response time) and the trust (i.e., the degree of compliance of a service to its specification) depend on the context of the services. Hence, when designing systems composed out of such services, it is important to consider the entire system execution context in addition to its functional and non-functional requirements. Achieving this goal requires the consideration of the context to trust-QoS dependencies, and interaction patterns between individual services. In this work, we tackle this challenge by first proposing a model that uses the context-QoS dependency information of individual services and inter-service interaction patterns to make predictions about the QoS and trust of compositions at the design phase. Then we apply the prediction model to select the optimum set of service to create the composed system. Our approach allows for better design and implementation decisions about composed systems in the early phases of the software lifecycle thereby reducing cost, time, and effort. The preliminary results show that the proposed model provides more accurate predictions and results in optimum composed systems than the prevalent approaches.

Keywords: QoS, trust, system composition, context.

1. INTRODUCTION

Making software services context-aware allows them to behave adaptively with the dynamics of their hosting environments. Distributed Systems composed out of such services are immensely valuable in applications such as the Internet of Things (IoT) [1], [2], [3]. The IoT has been predicted as a technology that will grow widely in the near future [4], [5]. As many of us start to depend upon such distributed systems, guaranteeing the trust and the QoS of these systems will be a major research challenge over the next few years. In this work, we propose a model that predicts the trust and the QoS of context-aware distributed systems in the early phases of the system development lifecycle.

In our work, the 'context' of the service/system is defined as the external factors that affect the QoS and the trust of the service/system. Since we are interested in the context and its effects on the QoS and the trust, for clarity, we further categorize the context of a service as following:

- **Physical context** - Physical attributes that affect the QoS. Examples are location, temperature, and humidity.
- **Association context** - Effects due to the presence of other services in the system on its QoS. For example, sharing the same Web session, or synchronizing the communication speeds with other services that affect the QoS of a service.
- **Input/Configuration context** - Effects of the input and the configurations of the services on its QoS. Examples are input data size (for a sorting service), or the resolution (for a camera service).
- **Execution context** - Effects of the hardware setup on which a service is running on its QoS. Examples are the size of the memory, and the processor speed.

Many prevalent techniques ([6], [7]) that evaluate QoS and the trust of service compositions not only lack the consideration of an explicit context in their evaluations, but also operate at later phases (such as testing or maintenance phases) of the system development lifecycle. These techniques are used to perform post-analysis of the software design, identify its faults, and improve the design in the subsequent iterations. The analyses provided by these techniques include:

- **Identification of the QoS/trust bottlenecks of individual services and interactions among services.**
- **Identification of the critical paths that affect the QoS/trust degradation or improvement in the overall system.**

In these analyses, artifacts from tests or deployment phases, such as the actual execution traces of the software systems, and end user experience, are used. However, the cost of fixing the issues found in later phases of the software life cycle, rather than in the early phases, increases exponentially with time [8].

Therefore, it is important to find techniques that would help developers to perform a similar analysis in the early phases of the system lifecycle. However, there are only limited amount of artifacts available in the early phases to carry out such an analysis. Example artifacts that are available in the design phase include the design diagrams, service specifications and execution traces of candidate
services, execution traces and user experiences about similar systems (that use a subset of the same candidate services), and the knowledge of the domain experts. It is challenging to carry out a useful analysis about the behavior, the QoS, and the trust of the composed system from this data. The techniques proposed in this work attempt to overcome this challenge by incorporating the context and its dependencies of the individual services, their interaction patterns, and the context of the overall composed system.

The rest of the paper is organized as follows: Section 2 compares between our trust prediction model against the state-of-the-art work; Section 3 describes the proposed trust prediction model; Section 4 discusses the application of our model using a case study; Section 5 describes an algorithm to apply the proposed model in the optimum service selection problem; Section 6 studies the behaviour of the optimization algorithm with large simulated data sets; Section 7 discusses the application of the optimum service selection algorithm to a real-world case study, and Section 8 presents concluding remarks with the future work.

2. RELATED WORK

There are many existing efforts that provide models to predict the QoS of composed systems at the design phase of its development lifecycle. A model provided by Jaeger et al. [9] uses QoS composition operators, which are based on the nature of QoS properties and the interaction patterns of the services, to calculate the QoS of compositions from the QoS of individual services. They have identified composition operators for common QoS attributes, such as the execution time, cost, encryption, and throughput, and common interaction patterns, such as sequence, loop, and different parallel compositions. However, these composition operators do not consider the impact of the environment and the correlations between services in calculating the QoS of a composition, which are factors that affect the QoS of context-aware services. Jaeger et al.’s model is extended by Hwang et al. [10] by including the composition operators that consider the distribution of QoS values for both individual services and the resulting predictions for the composed systems. Their approach also lacks the consideration of the environment and other external factors (context) in predicting the QoS of composed systems. In the subsequent sections, we show that their approach (which is referred to as the prevalent approach) gives lower accuracy compared with our approach when predicting the QoS of context-aware composed systems.

A similar operator-based approach is used by Elshaafi et al. [11] to predict the trust of compositions. They assume the trust of a composition can be predicted by identifying set of properties (which they call as trustworthiness properties) such as reputation, reliability and security. Similar to Jaeger et al. and Hwang et al., those properties are calculated using operators, which in turn depend on the nature of the property and the interaction pattern. In contrast to the other methods, their work include the operators for reputation, and shows a comparison of the use of such operators along with results of just averaging or taking the minimum among the reputations of participating compositions. They claim that as the trust changes with time, their approach shows the change better than the other approaches. However, they do not consider the context-QoS dependencies, therefore, their evaluations do not reflect the changes in trust due to the changes in the context.

Hang et al. [7] have used the mixture of beta distribution to represent the trust of services/compositions and predict the trust of the composition by considering dependencies of the composition with its participating services. These dependencies are represented using a Bayesian network. The beta distribution keeps track whether a service is trustworthy or not as a binary variable. Mehdi et al. [12] extend Hang et al.’s approach to use multinomial variable instead of a binary variable (which keep track of different trustworthiness levels) and the mixture of Dirichlet distribution to derive the composed trustworthiness. If the trustworthiness of the participating services is unknown, they propose an EM algorithm-based technique to infer the unknowns, which can be used to extract trust values of individual services from the existing systems. The main problem with these two approaches is that they validate their solutions using a synthetic dataset, which may not match with the real life situations. In addition, these approaches do not evaluate the trust associated with QoS parameters (which are mostly continuous variables) of the system and do not consider the effect of the context on the QoS in their evaluations.

There are other efforts that use machine-learning techniques to predict the QoS of compositions. For example, the model provided by Eskenazi et al. [13] uses linear regression to predict the performance of software systems that are composed of components. This requires the system developers to identify performance related parameters of each component manually, which are referred to as signature types, and use them as the features for the regression model while using the weights of each signature type as the parameters of the model. The regression model helps to extract out the importance of each signature type to the overall performance of existing systems and also provides ways to predict the performance of new systems. The main drawback of this approach is the need for an extensive manual effort in identifying signature types (which may not be shared among many services) and mapping them to adequate numerical values. These works also do not consider the context of the services in their performance predictions.

In our previous work [14], [15], we have used a subjective logic-based model [16] to represent the trust of services as a tuple of Belief, Disbelief and Uncertainty. We also proposed trust composition operators for basic composition patterns (mostly taken from the subjective
logic aggregation operators) that would aggregate the trust of individual services to derive the trust of composed system. Our results showed that the uncertainty component of the trust predictions is comparatively higher when empirically compared with the trust of the actual system. The reason for this observation is that the subjective logic operators are designed to be used with discrete variables, whereas the most of the QoS attributes are continuous values and the subjective logic is not fully capable of representing the trust of such variables. Therefore, in this paper, we represent the trust as probability values, which only have belief and disbelief components.

There are other efforts that study the context-QoS dependencies of services and find efficient algorithms to optimum service selection problem. For example, Mabrouk et al. [17] present an optimum QoS-aware selection algorithm that is fast enough to adjust to the changing context and re-evaluate the optimization problem with new QoS values (which have changed due to the context change). However, their focus is not to capture the context-QoS dependences of individual services or predict the QoS/trust of service compositions. The model provided by Guo et al. [18] shows that selecting correlated services for a composition would improve the QoS of the composed system. Similarly, the model provided by Barakat et al. [19] uses correlations between services to reduce the search space of candidate services while selecting the best subset of services for a particular functional requirement. Both the approaches ([18], [19]) consider the associativity context of the services in selecting services for a composition. However, their approaches cannot be extended to carry out an optimal service selection based on other context parameters (such as physical, input/configuration, and execution contexts). Additionally, there are many theoretical approaches ([20], [21], [22]) for finding efficient algorithms for the optimum service selection problem. These approaches validate their algorithms only using a simulated data set, and ignore the impact of the context and trust in their evaluation. In contrast, we have validated our approach using both simulated and real-life datasets and our approach consider the dependencies of the context to trust and QoS of services and systems.

### 3. QoS AND TRUST PREDICTION MODEL

Our trust prediction model consists of following four phases:

1) Collect the Context-QoS dependency information of individual services.

2) Collect information about the interaction pattern in the composed system.

3) Derive Bayesian networks for the context-QoS dependency for compositions.

4) Use inference techniques to answer relevant QoS/trust queries.

Each of the above phases is described with the help of a case study involving an Indoor Tracking System [23]. The tracking system is used to track positions of markers placed inside an indoor environment. The tracking system is created by composing a few atomic services. Table 1 lists the role of each of the participating services, service context values, and the QoS values.

#### 3.1 Phase 1: To collect the Context-QoS dependency information of individual services

To predict the trust of any future composed system, our framework needs the information about the context-QoS dependencies of each participating service. In addition to that, the framework keeps track of the context-context, and QoS-QoS dependencies as well. If a service (S) has QoS properties (Qs = q₁, q₂, ..., qₙ) and associated context properties (Cₛ = c₁, c₂, ..., cₙ), then there are functions fₓₛ and fₓₛₖ, such that

\[ fₓₛ : Cₛ → Qₛ \text{ for } y=1, 2, ..., m \]  
\[ fₓₛₖ : \{Cₛ, Qₛ\} → Qₛ \text{ for } x = 1, 2, ..., n \]

Here \( fₓₛ \) indicates the dependencies of the context and other QoS properties to a QoS property, whereas \( fₓₛₖ \) represents the dependencies between a context property to another context property. Note that, in this paper, we assume that both QoS and context properties can be represented as numerical values. If there are discreet numbers of possibilities for a context or QoS value, then each possibility can be represented as a binary variable.

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
<th>Context</th>
<th>QoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera Tracking Service (S₁)</td>
<td>Tracks the position of a marker relative to a smart phone camera.</td>
<td>Distance to tracked object, Angle to tracked object, Resolution width, Resolution height</td>
<td>Tracked error (in x direction), tracked error (in y direction), Net tracked error</td>
</tr>
<tr>
<td>Wi-Fi Tracking Service (Sₙ)</td>
<td>Tracks the position of a smart phone by triangulation of signal strengths from three Wi-Fi routers.</td>
<td>Distance to each Wi-Fi access points, Signal strength of each Wi-Fi access points</td>
<td>Tracked error (in x direction), tracked error (in y direction), Net tracked error</td>
</tr>
<tr>
<td>Average Fusion Service (S₁)</td>
<td>Fuses two or more independently predicted positions to derive an accurate position of the marker.</td>
<td>Input errors in x direction and y direction</td>
<td>Total error</td>
</tr>
</tbody>
</table>

Table I. Services in Indoor Distributed Tracking System
These context-QoS dependencies, context-context dependencies, and QoS-QoS dependencies of each service are represented in a Bayesian network [24]. A Bayesian network is a directed graph (Gs) with a set of vertices (V), edges (E) and dependency functions (F) such that

\[ f^\text{bn}: E \rightarrow V \rightarrow V \]  

\[ u = f^\text{bn}(v_1, \ldots, v_n) \]  

\[ \forall u \text{ where } (v_1, u) \in E, \ldots, (v_n, u) \in E \]  

Equation 4 indicates the relationship of each child vertex to its parent vertices \((v_1, v_2, \ldots, v_n)\) when there are directed edges \((v_1, u), (v_2, u), \ldots, (v_n, u)\).

We use the following mappings to translate the context-QoS, and context-context dependencies of a service to a Bayesian network.

- Each context property \(c_x^\text{cs}\) and each QoS property \(q_x^\text{qs}\) is mapped onto a vertex of the Bayesian network.
- There are directed edges either from a context vertex to a QoS vertex or from a context vertex to another context vertex.
- The function \(f^\text{bn}\), called as the dependency function, keeps track of the context-QoS, context-context, and QoS-QoS dependencies in a quantitative format. The function can be represented as either an algebraic function, or function of distributions (e.g., conditional linear Gaussian distribution) or as a tabular format.

![Figure 1. Bayesian network of the camera tracking service.](image)

A special advantage of using a Bayesian network is that it has a graphical representation that can be easily interpreted by humans in addition to manipulation by software applications. For example, for a camera tracking service \(S_2\), the QoS attributes \(Q_S^x\) are the response time \(rt\), and the tracked error \(e\). The associated context parameters \(C_s^x\) are the distance to the object \(d\), the angle to the object from a direction perpendicular to the camera face \(a\), resolution width \(w\), and the resolution height \(h\). The Bayesian network corresponding to the context-QoS dependencies of camera tracking service is shown in Figure 1.

The creation of the representation of a service’s context-QoS, and context-context dependencies in a Bayesian network requires the following two steps:

1) To evaluate the structure of the Bayesian network (Structure Learning).
2) To evaluate the dependency functions for each node of the Bayesian network (Parameter Learning).

To evaluate the structure of the Bayesian network (Structure Learning): The structure of the Bayesian network consists of the vertices \(V\) and the edges \(E\). This is mostly evaluated semi-automatically. The reason for that is even in the automatic evaluation of the structure (e.g., using likelihood scores [24]), there are many threshold parameters that should be decided by the human experts. Therefore, the structure of the Bayesian network has to be either evaluated or validated by a domain expert, which results in a semi-automatic approach to learning the structure of the Bayesian network. We have developed a graphical tool as an Eclipse plug-in [25] to ease the designing of the network structures for the domain experts (Screenshot of the tool is shown in Figure 2).

![Figure 2. Graphical editor to help designing of the structure of Bayesian networks.](image)

To evaluate the dependency functions for each node of the Bayesian network (Parameter Learning): Since the dependency functions have to be evaluated quantitatively, it is recommended to use automatic learning techniques (along with the help of a domain expert to do parameter tuning), than manual techniques. However, as most of the interested QoS attributes are continuous variables, it is important to find parameter learning techniques that can be used with continuous variables. In the case study, we have used the following three techniques.

- Using probability tables after discretizing the data [26].
- Regularized least squares regression [27].
- Bayesian Linear regression with sampling [27].

All these techniques require data points that are obtained from execution traces. Each data point should include the instance values of the QoS and context attributes.

To evaluate trust from the context-QoS Bayesian network: After modeling the Bayesian network for a service
(or a composed service), we will find the distributions of the QoS attributes under the context that service is actually going to operate in. Since trust is defined as the degree of compliance of the service to its specification, we will evaluate the trust corresponding to a particular QoS by calculating the portion that follows the specification from the distribution of the QoS. In our experiments, we present the trust corresponding to different possible specification values as it shows the errors of different approaches in predicting trust of the service/composed service more clearly.

### 3.2 Phase 2: To collect information about the interaction pattern in the composed system

In service compositions, services interact with each other to provide additional functionalities. Interaction patterns between services are selected by system designers to fulfill the system functionality requirements as well as to improve the QoS of the system. Primitive interaction patterns are Sequence, Split/Join, Exclusive Choice, Discriminator, Loop and Gateway as explained by Hwang et al. [10], and Gamage et al. [14]. Complex interaction patterns are made by recursively combining these primitive patterns.

**Composition Operators:** If a set of services is composed using interaction patterns, then the composed service aggregates the properties (i.e., both functional, and QoS attributes) of individual services. The resultant system properties depend on both the nature of the property and the interaction patterns used to aggregate the services. Formally, if a service \( S \) is composed of individual services \( S_1, S_2, \ldots, S_n \), then a property \( P \) of the service \( S \) is evaluated as a function of \( P_S \) of the individual services. This function is called the composition operator \( OP_{PS} \) and is defined as:

\[
OP_{PS} : \{P_S^1, P_S^2, \ldots, P_S^n\} \rightarrow P_S
\]  

### Table 2. The composition operators for different QoS properties and each interaction pattern. [14]

<table>
<thead>
<tr>
<th>Interaction Pattern</th>
<th>Response Time</th>
<th>Availability</th>
<th>Authentication &amp; Authorization</th>
<th>Confidentiality &amp; Integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>Addition</td>
<td>Multiplication</td>
<td>Universal</td>
<td>Universal</td>
</tr>
<tr>
<td>Parallel</td>
<td>Maximum</td>
<td>Multiplication</td>
<td>Universal</td>
<td>Universal/Generative</td>
</tr>
<tr>
<td>Split/Join</td>
<td>Exclusive</td>
<td>Non-deterministic</td>
<td>Universal</td>
<td>Universal</td>
</tr>
<tr>
<td>Exclusive Choice</td>
<td>Discriminator</td>
<td>Existential</td>
<td>Universal [with min resp time]</td>
<td>Universal</td>
</tr>
<tr>
<td>Loop</td>
<td>Multiplier</td>
<td>Exponential</td>
<td>Universal</td>
<td>Universal</td>
</tr>
<tr>
<td>Gateway</td>
<td>Addition</td>
<td>Multiplication</td>
<td>Universal [gateway]</td>
<td>Universal</td>
</tr>
</tbody>
</table>

### 3.3 Phase 3: To Derive Bayesian networks of context-QoS dependency for compositions

After building the Bayesian networks for individual services and identifying the composition operators, we derive the Bayesian network for the composed system using the following rules:

- Bayesian networks corresponding to the individual services will be part of the composed Bayesian networks as sub-networks.
- If two services share a context, then the composed Bayesian network will have a one vertex representing the shared context.
- There will be new QoS vertices for each interaction between services (corresponding to the composed QoS attribute), new edges connecting the new QoS vertices to other QoS attributes in the networks (that the new QoS depends on), and dependency functions corresponding to the composition operators.

**Deriving composed Bayesian network for a case study:**

As a simple case study, we compose a Wi-Fi tracking service and a camera tracking service with the sequence interaction pattern. The graphical representation of the composed Bayesian network is shown in Figure 3.

The total response time (\( tr \)), and the Total error (\( te \)) are the QoS attributes resulting from the inter-service interaction, become vertices in the composed Bayesian network. New edges are added from the QoS vertices, associated with individual services, to the new composed QoS attributes. For example, there are edges from \( r1, r2 \) vertices to the RT vertex. The composition operators are represented as the dependency functions associated with the new vertices of the composed Bayesian network. For example, addition operator is used with the RT vertex.

### 3.4 Phase 4: To use inference techniques to answer QoS/trust queries

After creating the Bayesian network for the composed system, the next step is to perform inferencing on the composed system. Following types of inferencing can be carried out:

1) To predict the QoS of the composition, if some of the context information is available using forward sampling.
2) To predict the trust of the composed system, if some of the context information is available using forward sampling.
3) To identify the context attributes to get the required QoS for the composed system using rejection sampling. The first two types of inferences can be used to select the most optimum set of services (when more than one candidate services are available) to create the composed system. An algorithm for optimum service selection problem is discussed in details in Section 5. The third type of inference can be used to evaluate the physical properties and the configurations of the production composed system to get the desired QoS and trust value after the deployment. For an example, in the composed tracking system (Figure 3), the tracking error and the response time can be improved up to the desired QoS by adjusting the configuration parameters such as the resolution width and height, as well as having lowering distances and angles between tracking objects and cameras. System developers can infer these information using our model to decide the quality and the quantify of the cameras required to build their system.

4. CASE STUDY AND RESULTS
VALIDATING THE MODEL

Results of learning and inferencing of the Bayesian network for a single service: In this section, we describe the building of the Bayesian network for the camera tracking service. For that purpose, we have used the domain knowledge of the cameras to identify the important QoS attributes associated with the service and the context parameters that would significantly affect these QoS attributes. These details are used to create the structure of the Bayesian network. The dependency functions of the network are learnt from the data observed from the execution traces using probability tables after discretizing the data, regularized least square regression and the Bayesian linear regression with sampling algorithms.

The structure of the Bayesian network for the camera service is shown in Figure 1. The data to train its Bayesian network are obtained from the actual execution traces of the services. We alter the source code of the camera service to log the QoS and the context values of the service. The actual position of the tracked object was manually inputted to the service, as the service was able to calculate the tracking error by subtracting the tracked position from actual position. The response time was calculated by measuring the time taken to process the video and track the position of the object. Each data point of the execution trace consists of values of context attributes (i.e., distance(d), and the angle(a) to the tracked object from the camera axis, resolution width(rw), and height (rh)), and the QoS attributes (i.e., the tracking error (e), and the response time (rt)). Our experiments generated 39061 of data points obtained by varying the context values and recording the corresponding data points.

The error values associated with the results of the inferences are calculated using the relative absolute error metric. [28]. The results obtained by using algorithms mentioned in section 3.1 are shown in the table 3. The table also shows the error for the predictions computed using the mean values of the training data, which assumes that the QoS parameters are independent of the context, therefore, no context dependencies have to be considered in predictions.

The above results show the impact of the context on the QoS attributes of the camera service. Lesser error values are obtained from the algorithms that consider the dependencies with the context than the techniques that do not consider the context.

The tracking error is sensitive to the context parameters, therefore, the context-aware learning techniques mentioned in Section 3.1 provide more accurate predictions. Since the response time is not strongly dependent on context, its predictions with those algorithms are much closer to the mean response time. We have used 0.05 as the p-value threshold in our experiments to address the issue of statistical significance. With this threshold, it is observed that the predictions, for both the tracking error and the response time, are significantly different from the mean values. The statistical significance results and the relative absolute error values shows the proposed approaches, specially the Bayesian linear Regression with sampling approach, provide significantly better predictions than the
Figure 4. The trust of the error using different approaches for different choice of specification values (for camera tracking service).

From the Figure 4 and Table 4, it is clear that if the trust is predicted without considering the context, then the prediction can be very different from the actual value when the system is run under a restricted context, whereas our approach that considers the context gives comparatively accurate predictions.

These results confirm that trust predictions that consider the context yield more accurate results than the prediction mechanisms that do not consider the context.

Inferencing of a Bayesian network of a composition for a case study: Here, we use the indoor tracking system as the case study to carry out similar inferences on the composed system. The architecture of the composed system indicating prevalent approach of using the mean value. From here on, we use Bayesian linear regression with sampling to perform additional inferencing such as trust evaluations on individual services and composed systems due to the higher accuracy of the technique.

The Bayesian linear regression predicts the set of values for a QoS attribute as a Gaussian distribution with a mean that depends on the context values. To evaluate the trust associated with the QoS for a particular context, we calculate the probability that the distribution in that context has adhered to the QoS specification. If the service is used in a context where the context attributes can take any values used in the training of the above prediction models, then the QoS distribution would be same as the QoS distribution of the training data. However, if the context is restricted, then it is possible to use the above prediction mechanisms to derive the distribution of the QoS under the restricted context. In our experiments, we restrict the context to the following,

1) Angle of the object (a) is restricted to $-0.25 \leq \tan(a) \leq 0.25$. We trained the model in the range $-0.5 \leq \tan(a) \leq 0.5$.

2) Distance to the object (d) is restricted to $120cm \leq d \leq 150cm$. We trained the model in the range $50cm \leq d \leq 600cm$.

We have selected the ranges mentioned here for the restricted context, due to the reason that the camera tracking service produces lesser noisy data in these ranges based on the empirical studies. With the restricted context, the trust values as a percentage for different tracking error specification values are listed in the Table 4.

Figure 4 shows the graphs for the predicted trust values without considering the context (prevalent approach), the predicted trust value with considering the context (our approach), and the actual percentage that met the specification against different choices of specification values.

Table 4. Trust evaluation with and without consideration of context for the camera tracking service

<table>
<thead>
<tr>
<th>Specification</th>
<th>3cm</th>
<th>4cm</th>
<th>5cm</th>
<th>6cm</th>
<th>7cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Trust (without considering context)</td>
<td>43.9</td>
<td>63.3</td>
<td>79.7</td>
<td>90.8</td>
<td>96.5</td>
</tr>
<tr>
<td>Predicted Trust (with considering context)</td>
<td>62.9</td>
<td>86.1</td>
<td>95.7</td>
<td>97.9</td>
<td>99.9</td>
</tr>
<tr>
<td>Percentage that actually met the specification</td>
<td>67.3</td>
<td>91.9</td>
<td>99.5</td>
<td>99.3</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3. Relative absolute errors of forward inferencing of single service Bayesian network

| Algorithm | Relative Absolute Error of Tracking Error (e|d,a,rw,rh) | Relative Absolute Error of Response Time (rt|rw,rh) |
|-----------|----------------|----------------|
| Using probability tables & message passing algorithm | 0.475 | 0.819 |
| Regularized least squares regression | 0.401 | 0.901 |
| Bayesian linear regression with sampling | 0.427 | 0.763 |
| From the means of training data | 1.070 | 1.040 |
the services and the interaction patterns between them is shown in Figure 4. We derive the Bayesian network of the
composition using the operators indicated in the Table 5 and perform inferences on it.

Table 5. Operators used in the case study

<table>
<thead>
<tr>
<th>QoS</th>
<th>Operators for Sequence pattern</th>
<th>Operators for Parallel Join pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time Error (X &amp; Y)</td>
<td>Addition</td>
<td>Mean</td>
</tr>
</tbody>
</table>

Table 6 shows the relative absolute error of the predictions obtained using the Bayesian network approach (which considers the context) and a prevalent approach [10] which does not consider context in their predictions.

Table 6 shows that considering the context gives more accurate predictions for the tracking error compared to the prevalent approach. However, the quality of the predictions for the response time has improved only slightly by considering the context. This is because the context does not have a major effect on the response time, where as it has a considerable effect on the tracking error. We have used 0.05 as the p-value threshold in our experiments to address the issue of statistical significance. With this threshold, it is observed that the predictions, for both the tracking error and the response time, of the two approaches are significantly different.

1) Camera Service 1: Angle of the object (a) is restricted to $-0.25 \leq \tan(a) \leq 0.25$. We trained the model in the range $-0.5 \leq \tan(a) \leq 0.5$.

2) Camera Service 1: Distance to the object (d) is restricted to $d \leq 100cm$. We trained the model in the range $20cm \leq d \leq 600cm$.

With this restricted context, the trust values for different tracking error specifications are visualized as a graph in Figure 6. The figure shows that our approach provides higher accuracy for predicting the QoS and the trust of composed services.

The application of the proposed model can be generalized to other domains. For such generalizations, a domain experts need to be involved in identifying the important QoS and context parameters in the first phase to obtain the structure of the Bayesian networks for each

![Figure 5. Design of the indoor tracking system](image)

![Figure 6. The trust of the error using different approaches for different choice of specification values (for the composed tracking system)](image)
individual service, and identifying the interaction patterns and the corresponding composition operators for the second phase. The resulting composed Bayesian network can be used to perform inferences specific to the domain. As the approximate inferences techniques that we used can be applied to large-scale Bayesian networks [29], we expect our model would perform equally well for complex systems also.

We have obtained better accuracy while predicting the QoS and the trust of systems by following the four phases indicated earlier. Our approach requires additional cost at the very early phases of the software lifecycle. However, as the predictions provided by our approach are significantly accurate than the prevalent approaches, the designers can make early decisions about the QoS and the trust of the end system, which would ultimately help them to save money, time and effort.

5. OPTIMUM SERVICE SELECTION PROBLEM

An important outcome that we can infer with the help of our model is to select the most optimum set of services for a composed system. When there are multiple services providing the same functionality, but with different QoS and trust properties, it is important to select the most optimum set of services that satisfy the QoS and the trust requirements of the composed system to obtain higher QoS and the trust. Naturally, the QoS properties of services compete with each other. For example, in a camera tracking service, the tracking error can be reduced by increasing the resolution configuration. However, that would increase the response time of the service. Therefore, it is important to find an optimization algorithm that would tackle such competing QoS properties, while satisfying the QoS and the trust constraints.

First, we formally define the problem as an optimization problem as follows.

Let there be ‘n’ service groups G1, G2, G3, . . . , Gm, in which the services in a same group provide the same functionality, and for each service group G, there are ‘m’ number of service implementations, S1, S2, S3, . . . , Sm. Each service (Si) has ‘p’ number of QoS parameters, qi1(C), qi2(C), qi3(C), . . . , qip(C). In practice, it is possible that not all the service groups have the same number of services and the not all the services have the same number of QoS parameters. We use those assumptions only for the sake of simplicity and without the loss of generality. Here the QoS parameters are functions of the context (C). (Here for the sake of simplicity, we have ignored context-context dependencies and the QoS-QoS dependencies). Our target is to select services from each group so that the following conditions are satisfied.

• Overall QoS and trust values satisfy the constraints requested by the user (satisfying feasibility constraints).
• QoS values have the optimum values within the user’s constraints (maximizing the objective function).

The feasibility constraints and the objective function would be determined by the system developer with the help of a domain expert. This optimization requirement can be modeled using the following multi-objective programing approach.

The binary variable Xij is used to indicate whether the Sij service is selected or not.

\[
X_{ij} = 1 \text{ if the service } S_{ij} \text{ is selected} \\
0 \text{ otherwise} \tag{6}
\]

Since we are selecting only one service from each group i, we can write set of constraints for feasibility as,

\[
x_{ij} = 1 \quad \text{for all } i = 1 \ldots n 
\tag{7}
\]

We have used the notation X to represent the set of Xij for all i, j values. Each instance of the X that satisfies the equation 7 represents a candidate composition.

Furthermore, our target composite service should satisfy the constraints enforced on each quality value. Assuming the user requirement for the kth QoS property is that its value should exceed Qk(C) with Tk(C) trust, feasibility constraints can be modeled using the following inequality equations.

\[
OP_{i=1..m} \sum_{j=1}^{m} x_{ij} q_{ij}^k(C) \geq T^k(C) Q^k(C) \text{ for all } k=1\ldots p \tag{8}
\]

Here the OPk represents the composition operator corresponding to the kth QoS property and the operator T^k(C) the inequality should hold with at least Tk(C) probability for a given feasible composition. Here, we assume that a higher QoS value for qij^k is desirable. If lower QoS values are desirable, then negative values should be assigned to qij^k to make sure to use the same inequality equation.

The left hand side of the equation 8 can be evaluated using our model discussed in section 3. It provides the probability distributions of overall QoS values for a composition of some selected candidate services. Then we would use these probability distributions to evaluate whether each QoS property has satisfied its required QoS and trust constraints or not. For a particular candidate composition (X), if the threshold of the kth QoS value that satisfies the trust requirement is denoted as Q^k_{T^k(C)}(X) the equation 8 can be rewritten as following.

\[
Q^k_{T^k(C)}(X) \geq Q^k(C) \text{ for all } k=1\ldots p \tag{9}
\]

The objective function ‘F’ would be the weighted average of the QoS properties. The weights are assigned by the users according to their preference of the QoS. Our optimization step is to maximize the following objective function.
\[ F = \sum_{k=1}^{p} W^k Q_k^{P}(C)(X) \]  

(10)

Zeng et al. [30] has shown a special case of the above multi-integer programming problem, in which the trust is not considered, the QoS does not depend on the context, and addition is the only composition operator allowed, is NP-complete. Therefore, the above problem (a more general problem than the problem mentioned in Zeng’s et al.’s paper) is also NP-complete. Therefore, in order to find an efficient solution to this problem, we would use a heuristic based optimization algorithm as indicated in the following sections.

We use the available context information from the deployment environment to identify the possible restricted QoS ranges of each services. That information will provide a more personalized optimum set of services for the system developer in addition to the personalization provided with the choice of the feasibility constraints and the objective function. After identifying the distribution of the context, we would use forward sampling on the context-QoS dependencies network to derive the corresponding QoS distributions. If no context information is available, then the expected values of the default QoS distributions are used in the optimization algorithm.

When evaluating the association context of services, we can identify the positive and negative associations between services in two different groups. For example, services that shares the same session information, and use the same protocols and technologies can have positive associations in terms of improving the response time, whereas services with different protocols and technologies can have negative associations and worsen the response time. In situations where we want to find the optimum set of services that improve the response time, we should favor the set of services that have positive associations that the set that has negative associations.

We have used an optimization algorithm based on the cross-entropy (CE) method [20], [31] with some improvements to include the proposed heuristic that captures the associations between services to identify the globally optimum set of services for a composition. The algorithm initially takes feasible samples of candidates compositions according to an initial probability distribution. Here, a sample contains a service from each service group. In each iteration, the probability values are adjusted by recalculating the objective function of the samples. Then, in the subsequent iterations, it produces more optimized samples with higher probability. The CE algorithm [20] is summarized in the following steps. The parameters \( p,d,\alpha \) mentioned in the below algorithm should be tuned to get to optimum solutions with higher objective values.

1) Assign uniform probabilities to all the services of each group. Say \( P_{ij} \) is the probability associated with the service \( S_{ij} \), then \( P_{ij} = 1/m \) for all \( i \) and \( j \).

2) In the \( t \)th iteration (initially \( t = 1 \)), pick \( N \) feasible samples, \( X_1, X_2, \ldots, X_N \). Each sample will have one service from each service group. The probability of picking a service \( S_{ij} \) from the \( i \)th group is \( P_{ij} \).

3) Sort \( X_i \) samples in the ascending order of the Objective Functions (\( F_i \)). Then, pick a value for the parameter \( p \) between 0 and 1, where last \( pN \) samples are considered relatively optimum samples. In the next steps, we would adjusts the \( P_{ij} \) values favoring these \( pN \) samples.

4) Calculate \( C_{ij} \) values for each service \( S_{ij} \) \( s.t. \)

\[ C_{ij} = \sum_{r=1}^{N} \frac{I_{F_{ij}>X_{ij}}}{N} \]  

(11)

where, \( \gamma = F_{(1-p)N} \) and

\[ I_{F_{ij}>X_{ij}} = \begin{cases} 1 & \text{if } F_{ij} > \text{for sample } 'r' \\ 0 & \text{otherswise} \end{cases} \]  

(12)

5) Update the \( P_{ij} \) values for the next iteration for each service \( S_{ij} \)

\[ P_{ij} = \alpha C_{ij} + (1-\alpha)P_{ij} \]  

(13)

6) If the \( \gamma \) values corresponding to the last \( d \) number of iterations are equal, then we can decide the solution is converged and the \( \gamma \) value can be used as the optimum objective function value. Otherwise repeat the step 2 for the \( t+1 \)th iteration.

With the above-mentioned heuristics, in capturing the associations between services, we expect to improve the algorithm by directing the algorithm to find a global optimum solution rather than a sub-optimum local solution.

We can improve the sampling process of services for candidate compositions in step 2 of the above algorithm by altering the sample probabilities to favor the candidate compositions that have more associations between services. For a particular sample, if we have already chosen services up to group ‘1’, then when we are selecting the service from the group \( i \) (\( i > 0 \)), we can evaluate the associations of the services in group \( i \) with the selected set of \( 1 \) services. We calculate the likelihood of the service \( S_{ij} \) being selected from the service group \( i \), when the services up to \( i-1 \) \( (s_1, \ldots, s_{i-1}) \) is already selected \( (L(S_{ij} | s_1, ..., s_{i-1})) \) using the following expression:

\[ L(S_{ij} | s_1, ..., s_{i-1}) \propto 1 + \frac{F_{\bar{S}_{ij},s_{i-1}}}{F_{\bar{S}_{ij},s_{i-1}} + |F_{\bar{S}_{ij}}|} \]  

(14)

Here, \( F_{ij} \) is the contribution of the service \( S_{ij} \) to the objective function (obtained by computing the weighted average of the QoS of the service using the same weights as the objective function), when the associations between the services are omitted. \( \Delta F_{ij} \) is the difference of the objective function from the service \( S_{ij} \) made due to the associations with the services \( s_1, ... s_{i-1} \). In this equation, if there are no associations between the already selected services and the service \( S_{ij} \), \( \Delta F \) becomes 0 and the likelihood
would be proportional to 1. If there is a positive association, then the $\Delta F$ would become a positive fraction, and the likelihood would be proportional to $L$, $1 < L < 2$. If there is a negative association, then the $\Delta F$ would become a negative fraction, and the likelihood would be proportional to $L$, $0 < L < 1$. This encourages the selection of a service with positive associations and discourage the selection of a service with negative associations. With this information, the posterior probability of selecting service $S_{ij}$ (denoted as $P(S_{ij}|s_1, ..., s_{i-1})$) can be evaluated using Bayes theorem, in which $P_{ij}$ is the prior probability.

$$P(S_{ij}|s_1, ..., s_{i-1}) = \mu L(S_{ij}|s_1, ..., s_{i-1}) P_{ij} \tag{15}$$

This expression has to be evaluated only for the services with associations as otherwise, the likelihood would be 1 and the posterior probability will be equal to the prior ($P_{ij}$). However, the posterior probabilities have to be normalized if any service is need to be updated due to the existence of associations. After that, the probability of the services ($P_{ij}$) of the service group is temporary updated to $P'_{ij}$ for the purpose of the sampling from that service group.

$$P'_{ij} = \beta P(S_{ij}|s_1, ..., s_{i-1}) + (1-\beta) P_{ij} \tag{16}$$

Here the $\beta_t$ is a parameter that would diminish as the iteration number($t$) reaches higher values. In our experiments, we have used $\beta_t = \beta/k$ where $k$ is a tuning parameter and $\beta_t = 0$ when $t \geq k$. That way, our heuristic will be effective at the early iterations to speed up the convergence and direct the solution to the desired global optimum value. After many iterations ($t \geq k$), the probabilities of the services will be trained towards the optimum solution, therefore, it is not required to be altered by the proposed heuristic method and save additional computational time.

After finishing the sampling process with altered probabilities, we continue with the steps of the algorithm up to step 4 (calculation of equation 11). Then, after we select the prominent $\rho$ N number of samples, we would re-evaluate the $P'_{ij}$ only for the services in the selected samples and set the $P_{ij}$ values to $P'_{ij}$. This encourages the selection of associated services in future samples.

6. SIMULATION STUDY

In order to test the validity of the algorithm with the proposed heuristic that capture the association between services, we applied the algorithm to a simulated data set. Simulations have been used in previous works [20], [21], [22] to validate the results of optimum service selection algorithms that do not consider associations between services. With the use of simulations, we can vary different parameters and identify the corresponding behavior of the algorithm. In simulation, we generated data for ‘n’ number of service groups and ‘m’ number of services for each groups. Each service has ‘q’ number of qualities (Q) and they depend on ‘c’ number of contexts(C). Q can be represented as a linear combination of C and its weights are generated randomly($w_C$). The selected set of contexts in ‘c’ are association contexts ($AC, AC \subset C$) and enables only when some other related services (service relations randomly are defined uniformly) are available in the same composition. These relations and the weights of the associations (both...
positive and negative) \( w_C \) are generated from a uniform random distribution.

The values for the simulation parameters are chosen to validate the scalability of the approaches in different practical scenarios. The parameters used for tuning the algorithm are chosen by experimenting with multiple runs containing different parameter values for a selected validation data set (obtained by simulations) and selecting the set of parameters that achieves better objective functions in less number of iterations. Objective function parameters, which are expected to be a users choice based on their requirements, are set to common values (giving all QoS parameters equal weights).

For the first experiment, we varied the \( w_{AC} \) from \( U(0,10) \) to \( U(100,110) \) and studied the effectiveness of the proposed heuristic as shown in the Figure 7.

From Figure 7, it is clear that when the associativity context is prominent, our heuristic method is effective in getting a comparatively higher objective value. Also, Figure 8 shows the algorithm has converged to an optimum value in lesser number of iterations. However, since sample probabilities are calculated in each samples, the runtime of the algorithm is improved only slightly with the use of the proposed heuristics as shown in Figure 9. Since, we focus on evaluating the compositions off-line at the start of software development lifecycle, we prefer algorithms that provides optimum solutions than the runtime efficiency of the algorithm. These graphs shows that the algorithm is capable of providing such solutions for systems with high association context dependencies.

Similarly, we have performed experiments by changing the number of service groups (Figure 10) and number of services for a service group (Figure 11) with lower and higher association context dependencies and evaluated the effectiveness of the algorithm. These graphs (Figures 10, 11) shows that the proposed heuristics find optimum solutions for higher association context dependencies consistently.

In Figure 10, the optimum objective function has increased with the increase of service groups (n). That is the expected behavior as in that situation the number of services per composition increases and each service contributes to provide higher QoS values for the composition. Furthermore, when the number of services per composition increases, the impact of the associations between services onto the composed QoS also increases. The results shows that the proposed heuristics has captured this associations more effectively and provided a higher value for the objective function than the traditional algorithm.

When the number of services per a service group (m) increases, the algorithm will have more choices to select the optimum set of candidate services for a particular composition. In such situations, we expect the algorithm to favor the selection of associated services to obtain the optimum solution. Figure 11 shows the effectiveness of the proposed heuristic based algorithm to select such associated services in getting a higher value for the objective function.

7. CASE STUDY - A TRAVEL PLANNING SYSTEM

We have applied our optimum service selection algorithm to a travel planning system. The service groups that the travel planning system is composed of and the corresponding candidate services are listed in the Table 7. The candidate services includes public services as well as custom written services. Some of the custom services are written to share the same SOAP session [32] with the services in other groups to emulate associations between them. We have used the same set of services in our previous work [15] to identify the association between public available services. In this case study we have altered the services as follows:
Each of the services is wrapped with a Web service that has four endpoints, in which, each endpoint is bound to a different Web service security policy to make them more secured, as mentioned below.

1) An endpoint with no security enforced.
2) An endpoint with username, password enforced (Usr).
3) An endpoint with signing of the messages enforced (Sign).
4) An endpoint with encrypting of the messages enforced (Enc).

Wrapper services are implemented and deployed using Apache Axis2 Web service engine [33] with Apache Rampart [34] and Apache Neethi [35] modules to provide the required security configurations. With these configurations, the Web services have three security related QoS properties, namely Authentication/ Authorization (Auth), Authentication/ Non-repudiation (Nrep), and Confidentiality (Conf), along with the non-security related QoS property, response time (Rst). For the Direction services and the Traffic services, the response time depends on the distance between the starting and ending places (Dist) of the input. As the distance increases, the message sizes also increase (with more directions and traffic information), the time to encrypt and sign the messages also increases. Furthermore, the response time also depends on whether the service is sharing the session with other services in a candidate composition or not. In our previous work [15], we observed that when two services from the same provider (e.g. MapQuest Direction Service and the MapQuest Traffic Service) are composed together sequentially, the aggregated response time is lower than the sum of the independently executed response times of individual services. Our conclusion is that the first service authenticates/authorizes the client and keeps the information in a shared session between services. Hence, the second service does not need to authenticate the client, thus, saving some computational time. We used the binary context parameter ‘Session Initiated’ (Sess) for each service to capture whether the corresponding session is started or not.

With these domain information, we are able to deduce the structure of the Bayesian network for the Direction Services and the Traffic Services as shown in Figure 12.
Table 9. The overall QoS values for optimum composition for different contexts

<table>
<thead>
<tr>
<th>Context</th>
<th>Predicted Rst</th>
<th>Actual Rst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr=True, Dist=200 miles</td>
<td>1666ms (DR1, TR1, HO3, WE2, CR1)</td>
<td>1837ms (DR1, TR1, HO3, WE2, CR1)</td>
</tr>
<tr>
<td>Usr=True, Dist=3000 miles</td>
<td>4160ms (DR2, TR1, HO3, WE2, CR1)</td>
<td>4025ms (DR2, TR1, HO3, WE2, CR1)</td>
</tr>
<tr>
<td>Sign=True, Dist=200 miles</td>
<td>2478ms (DR1, TR1, HO3, WE2, CR1)</td>
<td>2713ms (DR1, TR1, HO3, WE2, CR1)</td>
</tr>
<tr>
<td>Sign=True, Dist=3000 miles</td>
<td>6504ms (DR2, TR1, HO3, WE2, CR1)</td>
<td>6990ms (DR2, TR1, HO3, WE2, CR1)</td>
</tr>
<tr>
<td>Enc=True, Dist=200 miles</td>
<td>2797ms (DR1, TR1, HO3, WE2, CR1)</td>
<td>3013ms (DR1, TR1, HO3, WE2, CR1)</td>
</tr>
<tr>
<td>Enc=True, Dist=3000 miles</td>
<td>6537ms (DR2, TR1, HO3, WE2, CR1)</td>
<td>7069ms (DR2, TR1, HO3, WE2, CR1)</td>
</tr>
</tbody>
</table>

For the other set of services (HO, WE, and CR), the response time depends only on the security configurations. The structure of the Bayesian networks for the Hotel/Weather/Car Rental services is shown in the Figure 13. After deducing the structure of the Bayesian network, the parameters are trained for each individual candidate services independently using Bayesian linear regression.

As the services are composed sequentially, we would identify the corresponding composition operators for the interested QoS values using the Table 2. The operators are separately listed in the Table 8.

The Bayesian network corresponding to a candidate composition is derived by aggregating the Bayesian networks of the participating services as described in Section 3.3. Figure 14 shows an example of an aggregated Bayesian network. For simplicity, it shows only the ‘Session Initiated’ (Sess) context parameter and the ‘Response Time’ (Rst) QoS property. In this example, the two pairs (Direction service, Traffic service) and (Hotel service, Weather service) share the same sessions. In each session, the first service initiates the session, therefore, the ‘Sess’ will be set to true for the second service. The rest of the context parameters (Usr, Sign, Enc, Dist) are independently applied to the participating services. Furthermore, as the composition operator for the response time is an addition, we would arithmetically calculate the composed response time (by adding means and variance separately), instead of forward sampling on the composed Bayesian network.

In our experiments, we applied our algorithm for different contexts to find the optimum set of services and the corresponding overall QoS values for the travel planning system. We compared these results with: 1. an optimum set of services and the corresponding overall QoS values found using algorithm that does not consider context dependencies, 2. an actual optimum set of services and the corresponding overall QoS values found by executing all the possible combinations of candidate services.

The user security requirements decide the choice of the policy endpoint context and the feasibility constraints of the security QoS properties. For example, if the security requirement for the system is that it should support authentication/authorization, then the Usr context parameter is set to true, and the Auth is constrained to be true. Feasibility constraints for trust of the response time QoS property is set to 50% to work with mean of the probability distributions. As the feasibility constraints make sure the security QoS properties have desired values, the objective function is used to minimize the response time by assigning negative weight to the response time. Table 9 shows the overall QoS values for the optimum composition for different contexts using both the proposed algorithm and the actual executions of compositions. (In Table 9, only the ‘Rst’ is mentioned as the ‘Auth’, ‘Nrep’, and ‘Conf’ QoS properties can be implied from the context.)

From the results shown in Table 9, it can be observed that although the response time changes with the change in the Usr, Sign and Enc contexts, the optimum composition remain unchanged. That is because, we have implemented
the additional security layer as a wrapper to the existing services and the associated overhead is consistent among all the compositions. However, when the ‘Dist’ is changed from 200 miles to 3000 miles, the optimum composition also changes along with a rapid increase in the response time. We concluded the reason for this change is that when the distance is high, the Google Direction Service (DR1) sends a large output with more details whereas MapQuest Direction Service (DR2) sends a comparatively smaller output. For example, the message sizes for the direction between Indianapolis and Chicago (around 200 miles distance) are 32kb (DR1), and 14kb (DR2), where as the message sizes for the direction between New York and San Francisco (around 3000 miles distance) are 245kb (DR1), and 31kb (DR2). As the the services are wrapped with a security layer, the processing of the large output takes more time giving ‘DR1’ a higher response time compared to ‘DR2’.

If we use prevalent approach to predict overall QoS of the composition (without considering the context-QoS dependencies), the predictions for the two candidate optimum compositions are listed in Table 10. Therefore, if we had followed the prevalent approach, we would always select the composition that contains services DR2, TR1, HO3, WE2, and CR1 as the optimum composition regardless of the context it is being used. Whereas, if we used the proposed approach, we would use DR2, TR1, HO3, WE2, and CR1 services for compositions that deal with high distance travelling, and DR1, TR1, HO3, WE2, and CR1 services for compositions that deal with low distance travelling. Furthermore, the relative absolute errors of the prevalent approach and the proposed approach in predicting ‘Rst’ for the optimum compositions is compared in Table 11. As the ‘Rst’ strongly depends on all the context parameters, it is clear the consideration of such dependencies significantly improve the prediction error.

### Table 11. QoS predictions of the optimum service composition

<table>
<thead>
<tr>
<th>Approach</th>
<th>Relative absolute error of predictions of ‘Rst’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian network based approach (Considering the context)</td>
<td>0.204</td>
</tr>
<tr>
<td>Prevalent approach (Without considering the context)</td>
<td>0.949</td>
</tr>
</tbody>
</table>

8. CONCLUDING REMARKS

In this paper, we have proposed and empirically validated a model to predict the QoS and the trust of a composed system using the information available (such as context-QoS and context-context dependencies) at the design phase of the development lifecycle. We divide the evaluation process into four main phases: first building the Bayesian network of context-QoS dependencies for a single service; second, identifying composition operators for interaction patterns for each QoS; third, deriving the Bayesian network (of context-QoS dependencies) for the composed system, and the final phase, perform inferencing about the trust and the QoS of the composed system using the Bayesian network. We have presented the effectiveness of the proposed model by empirical validations using an indoor tracking system as a case study. Furthermore, we have presented how the optimum service selection problem can be tackled more effectively using the proposed model using a simulation and a real-world case study.

For future work, we plan to study what is the minimum required information about an existing system that will be helpful to get sufficient assessment (to be useful in modeling its QoS behavior) about a individual service, and what is the additional information that would improve the accuracy of the assessment. Furthermore, we are expecting to study fast Bayesian network inference techniques and fast optimization algorithms that can adapt to the rapid changes of the context. We also aim to validate the proposed solutions with more case studies from different application domains.

9. REFERENCES


Authors

Dimuthu U. Gamage: Dimuthu is a graduate student and currently reading for his PhD in computer science. Dimuthu’s current research focuses on quality aware and trust-based service composition and generative programming in distributed software systems.

Lahiru S. Gallege: Lahiru is a graduate student and currently reading for his PhD in computer science. Lahiru’s current research focuses on techniques for evaluating and validating trusted service representation, selection and negotiation in distributed software systems.

Dr. Rajeev R. Raje : Dr. Raje is a Full Professor and Associate Chair in the Department of Computer and Information Science at IUPUI. Dr. Raje’s research interests are in the fields of distributed computing, programming, component-
based software systems, and software engineering. Prof. Raje is a member of the ACM and IEEE.