12-2016

QoS and trust prediction framework for composed distributed systems

Dimuthu Undupitiya Gamage
Purdue University

QOS AND TRUST PREDICTION FRAMEWORK FOR COMPOSED DISTRIBUTED SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Dimuthu Undupitiya Gamage

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2016

Purdue University

West Lafayette, Indiana

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Dimuthu Undupitiya Gamage

Entitled
QoS and Trust Prediction Framework for Composed Distributed Systems

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Rajeev R. Raje Chair	Zhiyuan Li
Antony Hosking Co-chair	
Murat Dundar	
Mihran Tuceryan	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Rajeev R. Raje

Approved by: Sunil Prabhakar 12/06/2016

Head of the Departmental Graduate Program Date

To my parents, my wife and my little daughter.

ACKNOWLEDGMENTS

First, I would like to offer my gratitude to my research advisor Dr. Rajeev Raje for guiding me throughout the last six years with my research to make an impactful contribution to the the research community. I also like to thank the members of my advisory commitee, Prof. Tony Hosking, Prof. Zhiyuan Li, Prof. Murat Dundar, and Prof. Mihran Tuceryan for providing me constructive feedback about the research direction that I should follow.

I would like to thank the Security and Software Engineering Research Center (S2ERC) and the Air Force Research Laboratory (ARFL) for partially funding my projects. I would also like to thank the School of Science Graduate Student Council (SOSGSC) for providing me a travel grant to attend the ICWS 2015 conference.

I am thankful to Dr. William Gorman for helping me with the formatting of the dissertation and to Ms. Nicole Wittlief and Ms. Sandra Freeman for helping me with the examination procedures to meet the appropriate deadlines.

I am also thankful to IUPUI, the Office of International Affairs, and the Computer Science Department for providing me the opportunity to study and carry out research using the facilities, and for all the staff for their tremendous support.

Additionally, I like to thank all my collegues Lahiru, Manjula, Ruwan, Ryan, Aboli, and Amritha for sharing their experiences, and brainstorming ideas with me. I specially want to recognize Ryan, Aboli, and Amurtha for letting me use their own projects for my case studies to validate the proposed models. I feel very fortunate to have friends who support each other and celebrate each other's success.

Finally, I want to thank my parents, my wife, and my one-year-old daughter. My parents raised me providing the best education in my home country, heartfully blessed me to seek higher education in the United States and continue to show me the way to be a much better person. My wife dedicated her time and energy for our family with her never ending love, allowing me to focus on my research work and achieve my goals. My little daughter distracts me with her beautiful smiles, reminding me that the life is more than just work. I'm dedicating this dissertation work to all my family for their love, and energy that motivate me to carry out my work through all the obstacles.

PREFACE

This dissertation includes works that have been published in conferences and journals. Here, we present the publication list, and the chapters and the sections corresponding to each publication.

- Dimuthu U Gamage, Lahiru S. Gallege, James H. Hill, and Rajeev R. Raje. A compositional trust model for predicting the trust value of software system QoS properties. In Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering, pages 610617. IEEE Computer Society, 2012 **(Section 3.1)**.

- Dimuthu U Gamage, Lahiru S. Gallege, James H. Hill, and Rajeev R. Raje. Experimental evaluation of trustworthiness of compositional systems. International Conference on Network Infrastructure Management Systems (Interface 2013), Mumbai, India, 2013 **(Section 3.2)**.

- Dimuthu U Gamage, Lahiru S Gallege, and Rajeev R Raje. A QoS and trust prediction framework for context-aware composed distributed systems. In 2015 IEEE International Conference on Web Services (ICWS 2015), pages 4148. IEEE, 2015 **(Chapter 4 and Section 5.1)**.

- Dimuthu U Gamage, Lahiru S Gallege, and Rajeev R Raje. Composing context-aware distributed systems using QoS and trust principles. International Journal of Services Computing (IJSC), 4(2):3248, 2016 **(Section 6.1)**.

- Dimuthu U Gamage, Lahiru S Gallage, and Rajeev R Raje. A QoS and trust adaptation framework for composed distributed systems. In 2016 IEEE International Conference on Services Computing (SCC), pages 251258. IEEE, 2016. **(Section 6.2)**.

TABLE OF CONTENTS

Page

LIST OF TABLES ix

LIST OF FIGURES xi

ABBREVIATIONS xiv

GLOSSARY xv

ABSTRACT xvi

1 INTRODUCTION 1
- 1.1 Problem Statement 4
- 1.2 Motivational Case Studies 4
  - 1.2.1 Case Study 1: A Travel Planning System 4
  - 1.2.2 Case Study 2: An Adaptable Tracking System 6
- 1.3 Challenges 7
- 1.4 Proposed Approaches and the Contributions 11
- 1.5 Assumptions 14
- 1.6 Dissertation Organization 15

2 RELATED WORKS 16
- 2.1 Predicting QoS and Trust of Composed Systems 16
  - 2.1.1 Arithmetic Operators based QoS/Trust Predictions 16
  - 2.1.2 Machine Learning based QoS/Trust Predictions 19
- 2.2 Optimum Service Selection Problem 21
- 2.3 Dynamic Adaptations of Composed Systems 22

3 CONTEXT INDEPENDENT QOS AND TRUST PREDICTION FRAMEWORK 26
- 3.1 The Compositional Trust Model – BDUTrust 26
  - 3.1.1 Interaction Patterns of Properties 26
  - 3.1.2 Trust Operators and Representations 30
  - 3.1.3 Trust of Properties in a Composition 31
  - 3.1.4 Trust Composition Operators 32
  - 3.1.5 Mapping Composition Operators for Properties and Interaction Patterns 37
  - 3.1.6 Effect of Environment on Composition Operators 38
  - 3.1.7 Experimental Evaluation of the Compositional Trust model . 40
    - 3.1.7.1 Overview of the Indoor Tracking System 40

Page

3.1.7.2 Experimental Setup 42
3.1.7.3 Experimental Results of Compositional Trust model 45
3.2 Trust Composition Based on Regression Analysis – RegressionTrust 50
3.2.1 Machine Learning Model 52
3.2.2 Experimental Evaluation of RegressionTrust 54
3.3 Lessions 61

4 CONTEXT DEPENDENT QOS AND TRUST PREDICTION FRAMEWORK – CONTEXTTRUST 64
4.1 Phase 1: To Collect the Context-QoS Dependency Information of Services 65
4.1.1 To Evaluate the Structure of the Bayesian Network (Structure Learning) 69
4.1.2 To Evaluate the Dependency Functions (Parameter Learning) 70
4.1.2.1 Using probability tables after descreting the data . 70
4.1.2.2 Regularized least squares regression 72
4.1.2.3 Bayesian linear regression with sampling 73
4.1.3 To Evaluate Trust from the Context-QoS Bayesian Network 75
4.1.4 Collecting Context-QoS Dependency Information for the Case Study 76
4.2 Phase 2: To Collect Information about the Interaction Patterns in the System 78
4.2.1 Identifying Interaction Patterns and Composition Operators of the Case Study 79
4.3 Phase 3: To Derive Context-QoS Bayesian Networks for Compositions 81
4.3.1 Deriving Composed Bayesian Network for the Case Study . 82
4.4 Phase 4: Use Inference Techniques to Answer QoS/Trust Queries . 84

5 CASE STUDIES 86
5.1 Indoor Tracking System 87
5.1.1 Results of Inferencing on a Bayesian Network of a Single Service 87
5.1.2 Results of Inferencing on a Bayesian Network of a Composition 90
5.2 Travel Planning System 92
5.3 Collaborative Bullying Classification System 99

6 MORE APPLICATIONS OF THE QOS AND TRUST PREDICTION FRAMEWORK 111
6.1 Optimum Service Selection Model – OptimumTrust 111
6.1.1 Optimization Algorithm 112
6.1.2 Cross Entropy (CE) Algorithm 115
6.1.3 Ant Colony Optimization (ACO) Algorithm 118
6.1.4 Simulation Study of the Optimization Algorithm 120
6.1.5 Case Study – A Travel Planning System 126
6.2 QoS and Trust Based Adaptation Model – AdaptTrust 129

Page

6.2.1 Application of the ContextTrust Model at Runtime 131
6.2.2 Detection and Adaptation Services 132
6.2.3 Context Adaptation Algorithm 134
6.2.3.1 Continuous evaluation of the context 134
6.2.3.2 Evaluating the deviation of QoS and trust 136
6.2.3.3 Evaluating the adjustment of context 136
6.2.3.4 Recovering the context 138
6.2.4 Case Study – Adaptive Tracking System 138
6.2.4.1 Experiment 1: Tracking single marker 141
6.2.4.2 Experiment 2: Tracking two markers 142
6.3 QoS and Trust Evaluation Model for Reused Services – ReuseTrust 144
6.3.1 ReuseTrust Algorithm 144
6.3.2 Results of the ReuseTrust Algorithm 148

7 CONCLUSION 152
7.1 BDUTrust Model 153
7.2 RegressionTrust Model 153
7.3 ContexTrust Model 154
7.4 OptimumTrust Model 155
7.5 AdaptTrust Model 155
7.6 ReuseTrust Model 156
7.7 Lessons Learned 156
7.8 Future Work 157

REFERENCES 159

VITA 167

LIST OF TABLES

Table Page

3.1 The composition operators for different QoS properties and each interaction pattern – Part 1. 38

3.2 The composition operators for different QoS properties and each interaction pattern – Part 2. 38

3.3 Specification of response time and trust of services 43

3.4 Specification of error and trust of services 43

3.5 Predicted and actual values of properties and trust 50

3.6 Services used in travel planning system 55

3.7 Comparison of relative absolute errors of the results of models 57

3.8 Predicted values from the BDUTrust model 58

3.9 Predicted values form the RegressionTrust model 58

3.10 The actual values from empirical evaluations 59

3.11 Parameters related to service associations 59

4.1 Services in indoor distributed tracking system 65

4.2 Dependency functions for the Bayesian network in Figure 4.3 77

4.3 Dependency functions for the Bayesian network in Figure 4.5 78

4.4 Operators used in the case study for indoor tracking system 81

5.1 Relative absolute errors of forward inferencing of single service Bayesian network 88

5.2 Trust evaluation with and without consideration of context for the camera tracking service 89

5.3 Relative absolute errors of forward inferencing of single service Bayesian network 90

5.4 Trust evaluation with and without consideration of context for the WiFi tracking service 91

5.5 QoS predictions of service composition 92

Table Page

5.6 Trust evaluation with and without consideration of context for composed tracking system 93

5.7 Operators used in the travel planning system case study 96

5.8 Trust evaluation with and without consideration of context for travel planning system with ‘Travel-Context-A’ 98

5.9 Trust evaluation with and without consideration of context for travel planning system with ‘Travel-Context-B’ 98

5.10 Twitter data sources for each domain 102

5.11 QoS (PR,RC) of the classifier services for different twitter feeds 103

5.12 QoS distributions of the classifier services for different twitter feeds . . 104

5.13 Approximate composition operators for precision and recall 104

5.14 QoS of the ‘AND’ collaboration 105

5.15 QoS of the ‘OR’ collaboration 106

6.1 Default values for the simulation and algorithm tuning parameters . . 122

6.2 The overall QoS values for optimum composition for different contexts 128

6.3 QoS predictions of optimum compositions without considering the context 129

6.4 QoS predictions of the optimum service composition 130

6.5 Average response times of the system at triggers 142

6.6 Tuning parameters of ReuseTrust algorithm 148

LIST OF FIGURES

Figure Page

1.1 Abstract design of a travel planning system with candidate services . . 4

1.2 Setup of an adaptable tracking system 6

3.1 Sequence interaction pattern. 28

3.2 Parallel split/join interaction pattern. 28

3.3 Exclusive choice interaction pattern. 28

3.4 Discriminator interaction pattern. 29

3.5 Loop interaction pattern. 29

3.6 Gateway interaction pattern. 30

3.7 Illustration of the indoor tracking system setup. 42

3.8 Interaction patterns for the indoor object tracking system. 44

3.9 Non-deterministically select sensor services. 45

3.10 Sensor composed service. 47

3.11 Composed object tracking service. 48

4.1 Physical setup of the camera tracking service. 65

4.2 Physical setup of the WiFi tracking service. 66

4.3 Bayesian network of the camera tracking service. 69

4.4 Trust evaluation of the response time (rt) of camera tracking service of specification: $rt < 15msec$. 76

4.5 The Bayesian network corresponding to the context-QoS dependencies of WiFi Tracking Service. 78

4.6 Design of the indoor tracking system 79

4.7 Bayesian network of the composed service of WiFi tracking service and camera tracking service. 83

5.1 A graph showing the trust of the error using different approaches for different choice of specification values (for camera tracking service). . . . 90

Figure Page

5.2 A graph showing the trust of the error using different approaches for different choice of specification values (for WiFi tracking service). 91

5.3 A graph showing the trust of the error using different approaches for different choice of specification values (for composed tracking system). . . 93

5.4 Bayesian network of the direction/ traffic services 95

5.5 Bayesian network of the hotel/ weather/ car rental services 96

5.6 Bayesian network of a candidate composition 97

5.7 A graph showing the trust of the response time using different approaches for different choice of specification values (for travel planning system with 'Travel-Context-A'). 99

5.8 A graph showing the trust of the response time using different approaches for different choice of specification values (for travel planning system with 'Travel-Context-B'). 100

5.9 Bayesian network for the classifier services 101

5.10 Generic collaboration pattern of the bullying classification system . . . 102

5.11 A graph showing the trust of the precision for the data from political domain for different choice of specification values. 108

5.12 A graph showing the trust of the precision for the data from sports domain for different choice of specification values. 109

5.13 A graph showing the trust of the precision for the data from education domain for different choice of specification values. 110

6.1 Objective function vs the association context dependency weight 121

6.2 Algorithm iteration vs the association context dependency weight . . . 123

6.3 Algorithm runtime vs the association context dependency weight . . . 124

6.4 Runtime vs the association context dependency weight for CE, CE+Heuristics algorithms 125

6.5 Objective function vs number of service groups 126

6.6 Objective function vs the number of services in a service group 127

6.7 The high level architecture of the AdaptTrust model 132

6.8 Bayesian network of the camera adaptation service. 133

6.9 Evolution of the context with time with the Markov assumption. 136

Figure Page

6.10 Physical setup of the adaptive tracking system. 139

6.11 Services and their interactions in adaptive tracking system. 139

6.12 Actual vs tracked positions and points of triggers. 141

6.13 Actual vs tracked positions for two markers. 143

6.14 An application of 'ReuseTrust' algorithm 145

6.15 The mean absolute error of the μ_{rt} with the number of unknown services 149

6.16 The mean absolute error of the μ_{rt} with varying number of compositions 150

ABBREVIATIONS

SOA	Service Oriented Architecture
QoS	Quality of Service
IoT	Internet of Things
CPS	Cyber-Physical Systems
SOAP	Simple Object Access Protocol
NB	Naive Bayes
LR	Logistic Regression
SVM	Support Vector Machine
ACO	Ant Colony Optimization
CE	Cross Entropy

GLOSSARY

BDUTrust	Prediction model based on subjective logic
RegressionTrust	Prediction model based on association context
ContextTrust	Prediction model based on general context
OptimumTrust	Optimization model for service selection
AdaptTrust	Adaptation model for QoS and trust aware adaptations
ReuseTrust	Machine learning model to decompose QoS and trust

ABSTRACT

Undupitiya Gamage, Dimuthu PhD, Purdue University, December 2016. QoS and Trust Prediction Framework for Composed Distributed Systems. Major Professor: Rajeev R. Raje.

The objective of this dissertation is to propose a comprehensive framework to predict the QoS and trust (i.e, the degree of compliance of a service to its specification) values of composed distributed systems created out of existing quality-aware services. We improve the accuracy of the predictions by building context-aware models and validating them with real-life case studies. The context is the set of environmental factors that affect QoS attributes (such as response time and availability), and trust of a service or a composed system. The proposed framework uses available context-QoS dependency information of individual services and information about the interaction patterns among the services to make predictions for the QoS and trust values of the composed system at the design phase of the development lifecycle. Such predictions made in the early phases of the system development lifecycle will reduce cost, time, and effort. We demonstrate the use of these predictions in selecting the optimum set of services to create composed systems using heuristic optimization algorithms. Additionally, the prediction model is used at runtime with fast heuristic techniques to build adaptable composed systems. The empirical results show the proposed context-dependent framework performs well in providing more accurate predictions than the prevalent approaches.

1 INTRODUCTION

With the high availability of public software services, and the increasing tooling support, Service Oriented Architecture (SOA) [1] has become a common design and development methodology in developing software systems (especially distributed software systems). In SOA, the services, which are software with a specific functionality, are developed by experts from that particular domain. These services are designed to be both interoperable and reusable. Therefore, by adopting SOA, software system developers are able to implement the end system requirements rapidly and efficiently by composing functionalities of many services.

Typically, each service has a set of Quality of Service (QoS) properties in addition to having a specific functionality. Prominent examples of QoS properties are response time, availability, and reliability. In practice, a similar functionality can be provided by many service providers, but with different QoS properties. For example, there are many weather forecasting services (that are provided by different providers), and each service can have different response time, availability, and reliability values. Service providers publish the QoS properties along with functionality of the service in the service specification.

Trust is an another property associated with individual services and composed systems. Trusted Computing Group defines 'trust' as the following: 'An entity can be trusted, if it always behaves in the expected manner for the intended purpose' [2]. We have adapted this definition in the context of software services as the following: 'Trust of a service is the degree of compliance of the service to its specification.' Each entry of the service specification (i.e., functional and QoS property) has an associated trust value indicating the degree of compliance of the actual value to the specified value. To measure trust, we use both subjective evidences (e.g., end user ratings, and comments), and objective evidences (e.g., execution traces). We use two metrics for trust measurements in our work. Mainly, we use the probability of the compliance (i.e., a value between 0 and 1) as a metric for trust measurements. As an example, consider the specification of an object position tracking service, which can list its response time to be less than 30ms with 0.98 trust, and the tracked error to be less than 10cm with 0.95 trust. Such trust values are required (that the provider cannot specify a deterministic value for QoS) due to both the internal effects such as non-deterministic logic of the software and the external effects such as the changes in the execution environment of the software. Secondly, we use subjective logic [3] as another trust metric which is based on the uncertainty principles from the Dempster-Safer theory [4]. With that metric, an end user of a service/ system can express the trust subjectively from his experience as a tuple of Belief (B), Disbelief (D), and Uncertainty (U). For example, a user can express from his experience, the tracker error is within the satisfied level with (B,D,U) values (0.8, 0.1, 0.1). Here the 'B' and 'D' is analog with the probability of success and failure, whereas 'U' stands for the ignorance obtained due to the lack of complete details. In Chapter 3.3, we discuss the reasons for choosing the probabilistic representation of Trust in the majority of our work.

We refer to the environmental factors that affect the QoS and trust of the service/ system as the 'Context' of the service/ system. We are interested in the context and its effects on the QoS and trust values; hence, we further categorize the context of a service as the following:

- Physical context – This contains the set of physical attributes that affect QoS of a service. A few common examples are location, temperature, and humidity.
- Associativity context – This indicates the effect due to the presence of other services in the system on the QoS of a service. For example, sharing the same web session, or synchronizing the communication speeds with other services would affect the QoS of a service.

- Input and Configuration context – This represents the effects of the input and the associated configurations on the QoS of a service. Examples are input data size (for a sorting service), and resolution (for a camera service).
- Execution context – The effect of the hardware that the service is running on to its QoS. Examples are size of the memory, and processor speed.

Making services context-aware allows them to behave adaptively with the dynamics of their environments. Distributed Systems composed out of such services are immensely valuable in applications such as the Internet of Things (IoT) and Cyber-Physical Systems(CPS) [5–7]. The IoT and CPS have been predicted as the technologies that will explode in the near future [8–10]. 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 dissertation, we focus on helping the system developers to predict the values for the QoS and Trust aspects of distributed composed systems. We study the predictability of QoS and Trust of the service with different trust models such as context independent models and context dependent models. Our analysis shows that consideration of the context provides more accurate predictions with lesser uncertainty. We develop a context dependent trust prediction framework that will help the developers to infer QoS and trust properties of future systems in some user context using the available data at the design time such as the domain knowledge, individual system properties, interaction patterns of services, and existing similar systems. We demonstrate the application of the proposed framework to select the optimum set of services that will build QoS and trust optimized composed systems. Additionally, we further improve the framework to continuously monitor the changes in context after the services/ systems are deployed in production and makes them adaptive in real-time when QoS and Trust drops below the required thresholds. We empirically validate the proposed frameworks and its applications using case studies from indoor tracking, travel planning, and distributed bullying detection domains.

1.1 Problem Statement

Our objective is to build a comprehensive framework for accurately predicting the QoS and Trust values of distributed systems composed of individual services during both the design phase and the execution phase of the system lifecycle.

1.2 Motivational Case Studies

We present case studies from two domains to show the importance of the proposed QoS and Trust prediction framework in developing distributed systems. The case studies are:

1. A travel planning system

2. An adaptive tracking system

1.2.1 Case Study 1: A Travel Planning System

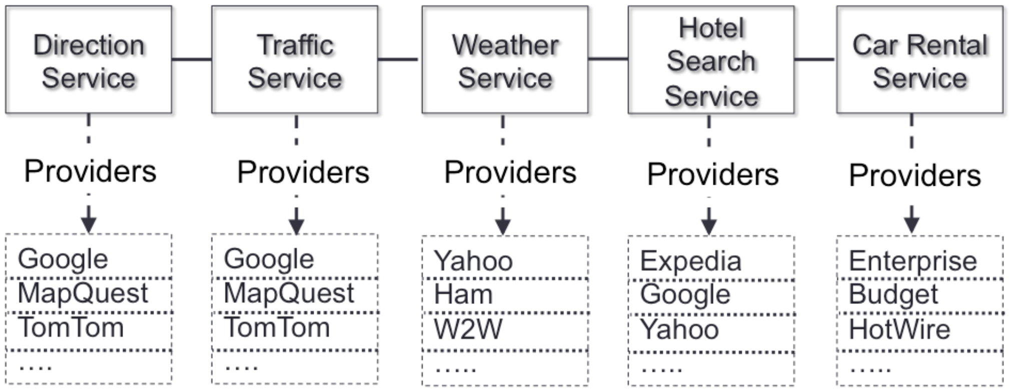


Figure 1.1.: Abstract design of a travel planning system with candidate services

The functionality provided by the travel planning system (shown in Figure 1.1) includes providing directions, traffic, weather, hotel search and car rental information for planning a travel itinerary. Prominent QoS properties of the system are response time, confidentiality (by encrypting information), availability and cost. Example context properties that affect these QoS properties are distance to the destination, which affect the response time of retrieving direction and traffic information, and the type of travel such as business or personal, which affects the confidentiality of the information and the associated cost.

When developing a travel planning system, the system developer first decides the categories of services (referred as abstract services) that should be included in the system to satisfy the functional requirements of the system user. Let us assume the included abstract services to be direction, traffic, weather, hotel search, and car rental services. Then the developer needs to select specific concrete services for each of these abstract service categories from the available candidate services that are provided by different vendors. Such decisions should consider the following factors:

- The system should satisfy user requirements of QoS and trust values. An example of a QoS and trust requirement is that system should have response time $< 20ms$ with 90% trust.

- The system should operate in a specific context as indicated by the user. An example of a context requirement is that the system should support long distance traveling.

- The system should deliver optimal QoS and trust values (while satisfying above user requirements). For example, the system should deliver lowest response time and cost, and highest availability.

The major challenge faced by such a system developer is that she has to make these decisions at the design phase of the development lifecycle. If she has to change these design decisions during the later phases of the development lifecycle, it will result in high cost. Another challenge is that these decisions depend on the context of the system, as any change in the context of a service will result in different values of QoS and trust than the advertised ones. Therefore, there is a need for a model that will help the developer to make appropriate design decisions related to QoS and trust values. The models proposed in this dissertation will address these challenges by providing high accurate predictions about QoS and trust values of composed systems and assist in selecting optimum candidate services to build quality and trust-aware distributed systems.

1.2.2 Case Study 2: An Adaptable Tracking System

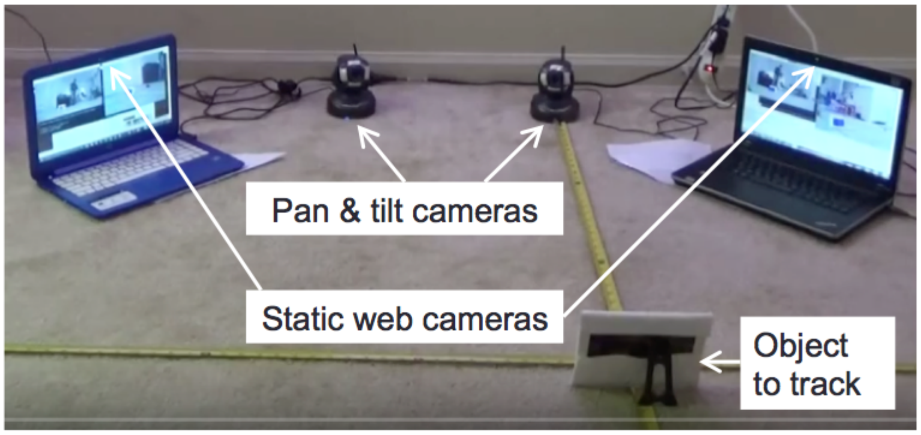


Figure 1.2.: Setup of an adaptable tracking system

The adaptable tracking system used in this case study is composed of two pan and tilt cameras (dynamic-cams), two static web-cams, and a fusions service. This system continuously tracks the positions of objects. Important QoS properties of the system are the response time and the tracking error. The context properties that affect the QoS properties are the distance and the angle of the object from the cameras, which affect the tracking error, and the resolution of the cameras, which affect both the response time and the tracking error. When objects are moving, the system triggers pan and tilt operations of dynamic-cams to keep the objects in tracking range of the cameras.

The major challenge of developing such a system is whether it is possible to keep QoS properties (e.g., tracking error) and the associated trust value of the system intact with the movements of the objects. If there is a degradation in the QoS and trust values, the system should self-adapt to minimize the damage and preferably keep them within a given QoS and trust threshold value. Additionally, it is challenge to make the self-adaptation process fast enough to keep its functional operations at a desirable level. The model proposed in this dissertation addresses these challenges by providing fast and accurate predictions of the QoS and trust values with the changes in the context, allowing the system to trigger necessary adaptations accurately and quickly.

In the next sections, we explore each of these challenges in detail, and how we address them using the proposed approaches.

1.3 Challenges

1. **Predicting QoS and trust of a composed system from the data available at the design time.** When developing software systems following the correctness by construction design principle, the system developers focus on achieving correctness of the system (in functionality, QoS, and trust aspects) from the very early phases of the system lifecycle. Otherwise, if defects (in any of the functionality, QoS, and trust aspects) are found in the later phases of the system lifecycle, it takes more cost, effort, and time to fix them [11]. That leads to missing deadlines, going over-budget and even failures of projects.

   Many prevalent techniques [12, 13] that evaluate QoS and trust of service compositions operate on later phases (such as testing or maintenance phases) of the system lifecycle. These techniques can be used to perform a post-analysis of the software design, identify its faults, and improve the design in the subsequent iterations. Their analysis includes,

   - Identify QoS/ trust bottlenecks of individual services and interactions among services.

- Identify critical paths that affect the QoS/trust degradation or improvement in the overall system.

They use artifacts from tests or execution phases such as actual execution traces of the software systems, and end user experiences in performing these analyses. However, as indicated above, fixing the issues found in later phases of the software life cycles costs lot higher (and grows exponentially with how later it is) than fixes issues found in early phases [11].

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

2. **Capturing the context-QoS/ trust dependencies of the context-aware services and composed systems.** QoS and trust values of the service can vary with the changes in its context. For example, in an object position tracking service that uses a camera images to track the position of an object, the tracked error and the associated trust can be vary with the change in the angle and the distance between the object and the camera. Therefore, QoS and trust prediction model should be able to capture the context-QoS/ trust dependencies from the evidences available at the prediction time. However, there can be many possible physical, input/ configuration, and execution context variations and

exponential amount of associations between participating services. Therefore, it is a challenge to track all the possible dependencies from context to QoS and trust quantitatively and use them to achieve useful QoS/trust inferences about the services and systems. To address this challenge, we use a Bayesian network based technique and continuous probability distributions to capture these dependencies and sampling-based techniques to carry out inferences on these networks.

3. **Identifying the optimum set of services to achieve the best QoS and trust values for the composition in a certain context.** When developing software systems by composing many individual services, the system developers try to select the most suitable set of services that satisfy their functional, QoS and trust requirements [14]. Since different QoS properties compete with each other (for an example, in an object position tracking service the QoS properties, tracking error and the response time, are competing with each other as the use of higher amount of processing to minimize the tracking error would also increase the response time) and different services are designed to optimize different QoS properties, it is not possible to select the best set of services considering only the individual QoS and trust values and expect the composition to have the optimum QoS and trust values. Selecting the optimum set of services for a composition is a multi-objective multi-variable integer programming optimization problem, which is shown to be an NP-complete problem [14]. That makes solving this problem a time consuming task, specially when there are lot of candidate services available. There are many prevalent works [14–16] proposing approximate heuristic algorithms to solve problem with a less time complexity. However, none of them have considered two important aspect of the problem, i.e., the optimization of trust and satisfying trust constraints, and the dependencies between the context and the QoS. It is a challenge to model this optimization problem and provide a heuristic algorithm that would solve the problem efficiently. In our work, we propose solutions addressing these

challenges based on a heuristic based optimization algorithm and validate these solutions empirically with real-life case-studies.

4. **Adaptation of services and composed systems with the changes in context at runtime.** The proposed QoS and Trust prediction model is capable of capturing the context-QoS dependencies of participating services and the composed systems using Bayesian networks. By inferencing on the Bayesian networks, the model can predict the QoS and trust values of the composed system for the context variations in early phases of the system development lifecycle. These predictions help the system developers to make better design and implementation decisions early in the design phase and integration phase of the system development lifecycle. However, when such systems are deployed in contexts that change often and rapidly such as the applications in IoT and CPS domains, it is hard for developers to design for all the anticipated variations of the context in advance. Therefore, it is important that the model can be applied to systems at runtime and continuously predict the QoS and trust values as well as improve the prediction accuracy at runtime. However, it is a challenge to evaluate Bayesian network models at runtime with the overhead of inferences techniques. To address this challenge, we develop heuristic-based fast inference techniques specially designed for the adaptation of services and systems. Additionally, our QoS and Trust adaptation model based on the QoS and Trust prediction model to feed data to the system about the changes in its context, and information about when the adaptation should trigger to keep the QoS and trust values of the system within satisfactory level. Therefore, the proposed Trust adaptation model can be used to make existing services and systems, which were not originally designed to be adaptable, to adapt with the changes in context and subsequent changes in QoS and Trust.

5. **Capturing the context-QoS dependencies of services from existing systems and use this information in predicting future systems.** As our

prediction model requires the context-QoS dependencies information of each services, its important to identify these information quantitatively to get highly accurate predictions. To capture these dependences, the services have to be run under different possible contexts and monitor its QoS. However, it may not be possible to test each individual services under all the possible contexts, specially for different associative contexts where the services have to be tested how it is interacting with other services.

Since the services are reusable, a service that contribute to a future system may have been used in existing systems. Since these systems can be executed in different contexts, we can extract out the context-QoS dependencies of the individual services by analyzing the execution data from these existing systems. However, its a challenge to extract out such details from a complete systems, as systems are made of many services with different types of interactions under different contexts. In our proposed approach, we expect to tackle this challenge by using both automatic and manual techniques to learn the service parameters quantitatively.

1.4 Proposed Approaches and the Contributions

We propose several models (BDUTrust model, RegressionTrust model, ContextTrust model, OptimumTrust model, AdaptTrust model and ReuseTrust model) to address the research challenges mentioned in Section 1.3. In this section, we provide an overview, inputs, outputs and summary of contribution of each model.

1. **BDUTrust model**: The BDUTrust model predicts the trust values of composed systems using subjective logic operators. We study the use of subjective logic to capture both subjective and objective evidences in evaluating QoS and trust values of both the individual services and the composed systems. The inputs to the model are the execution traces and the user experience (represented in the subjective logic representation) of individual services, and the

knowledge of domain experts about the interaction patterns present in the system. The output of the model is the trust prediction value of the composed system. Our contributions include: methods for computing B, D, U (subjective logic representation of trust) tuples of individual services using execution traces, identifying trust composition operators (mostly taken from the subjective logic aggregation operators) for basic composition patterns, prediction of trust of composed systems, and empirically validating the proposed approach using real-life case studies. In BDUTrust model we do not consider the importance of context in evaluating trust of the individual services and the composed systems.

2. **RegressionTrust model**: The RegressionTrust model predicts the QoS and trust of a particular, to be constructed, distributed systems by capturing the associations present between participating services in other related systems. The inputs to the model are the execution traces and the interaction patterns of existing related systems. The output of the model is the QoS and trust predictions of the future composed systems. Our contributions include: devising a machine learning based technique to identify associations between services quantitatively, and the use of QoS/trust values of individual services and associations between services to predict the trust of future systems. In the RegressionTrust model we only consider the importance of the association context in evaluating trust of the individual services and their composed systems.

3. **ContextTrust model**: The ContextTrust model predicts the QoS and trust values of the composed systems by considering the context to QoS and trust dependencies of individual services. We use Bayesian networks and associated learning and inference techniques to capture and infer the context-QoS/trust dependencies of the individual services and the composed systems. The inputs required to train the model are the execution traces of services, and the knowledge of domain experts about context-QoS dependencies, and interaction

patterns. The inputs required to infer the model are user context information. The output of the model is the QoS and trust prediction values of the composed system. Our contributions include: the capturing the context-QoS dependencies quantitatively from the execution traces of the services using Bayesian networks, aggregating these Bayesian networks of individual services to derive the context-QoS dependency Bayesian network of the composed system, studying different learning and inference techniques of the networks, providing more accurate predictions of the QoS and trust values of the composed systems when compared with the prevalent approaches and empirically validating the model with real-life case studies.

4. **OptimumTrust model**: The OptimumTrust model selects the optimum set of services to build QoS and trust optimized distributed systems. The inputs to this model are the candidate services of each abstract service category, the user context, the QoS/ trust constraints, and the QoS/trust preferences. The model outputs the optimum set of candidate services to compose the required distributed system. Our contributions include: modeling the service selection problem for context-aware distributed systems, identifying heuristic techniques to solve the optimization problem efficiently, and validating the efficiency of the proposed heuristic solution using simulated data with large number of service groups and candidate services.

5. **AdaptTrust model**: The AdaptTrust model helps the composed distributed systems to adapt with the changes in context during their execution. The inputs to the model are the readings from detector services about the context. The output of the model is in form of triggers to the adaptation services with quantitative information about the required adaptation level. Our contributions include: the study of heuristic techniques to speed up the inferences of context-QoS dependency Bayesian networks, adaptation model that keeps the QoS and

trust values within the specified thresholds with the changes in the context at runtime, and validation of the model using a real-life case study.

6. **ReuseTrust models**: The ReuseTrust model infers the context-QoS/trust dependencies of participating services using the context-QoS dependency information of their composed systems. The QoS and trust values of existing systems can be evaluated using evidences such as the execution history of the systems. However, the execution history information may not be available to be used for evaluating trust of individual services that the systems are composed of. The reason for that is that the users interface with the systems as a whole rather than with individual services. In such situations, this model can be used to infer QoS distributions of individual services. The inputs to the model are the context-QoS dependency Bayesian networks of composed systems, and the context-QoS dependency Bayesian networks of known services. The outputs of the model are the context-QoS dependency Bayesian networks of unknown services. Our contributions include: building a learning model to infer context-QoS information of individual services using context-QoS dependency Bayesian networks of multiple compositions, and validating the model using a real-life case study.

1.5 Assumptions

The proposed context dependent QoS and trust prediction framework can be used to make predictions about a composed distributed system. It operates under the following assumptions.

- The abstract design of the system (which consists of abstract services, and the interaction patterns among services) is already known. The abstract design of the system would be prepared to satisfy its functional requirements. Concrete services corresponding to each abstract service can be selected (from the set of candidate services) based on the QoS and trust requirements using the proposed model in this dissertation.

- For each candidate service, either the complete context-QoS dependencies should be known or execution logs of existing systems that have used the candidate service should be available, which will help in deriving the context-QoS dependencies of the service.

1.6 Dissertation Organization

This chapter introduces the problem statement of this dissertation, related challenges, summary of the contribution and the assumptions made by us in addressing these challenges. Chapter 2 discusses the existing works that have attempted to address the same challenges, and shortcomings of these attempts. Chapter 3 discusses the context independent QoS and trust prediction framework with the results of the empirical validations. Chapter 4 discusses the context dependent QoS and trust prediction framework in detail. Chapter 5 presents the experiments done to empirically validate the QoS and trust prediction framework and along with the results of these experiments. Chapter 6 discusses the applications of the trust prediction framework to real world problems with a case study for each application. Finally Chapter 7 concludes the dissertation with lessons learnt, inferences derived and possible further extensions.

2 RELATED WORKS

In this chapter, we analyze the related works in three categories mentioned below. These works attempt to address the same challenges mentioned in section 1.3. We will compare these efforts with our approach. The content in this chapter is an extension of our previous publications [17–21].

1. Predicting QoS and Trust of Composed systems
   (a) Arithmetic operators based QoS/Trust predictions
   (b) Machine learning based QoS/Trust predictions
2. Optimum service selection problem
3. Dynamic adaptations of composed systems

2.1 Predicting QoS and Trust of Composed Systems

There are two sub-categories of works that provide models to predict the QoS and trust of composed systems at the design phase of the system development lifecycle. The first sub-category of works uses arithmetic operators, and the second sub-category uses machine learning techniques to perform predictions. We discuss these two sub-categories of works separately.

2.1.1 Arithmetic Operators based QoS/Trust Predictions

A model provided by Jaeger et al. [22] uses QoS composition operators, which are based on the nature of the QoS property and the interaction patterns of the services, to calculate the QoS of compositions from the QoS of individual services. They have identified QoS composition operators for common QoS attributes, such as execution time, cost, encryption, and throughput, with common service interaction patterns, such as sequence, loop, and different parallel compositions. However, the proposed QoS composition operators do not consider the impact of the environment and the correlations between services in calculating the QoS of a composition. Jaeger et al.'s model is extended by Hwang et al. [23] by including QoS composition operators that consider the distribution of QoS values for both individual services and resulting predictions of 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 chapter 5, we show that their approach (which is referred to as the prevalent approach) gives lower accuracy compared with our approach, when predicting the QoS values of different composed systems.

Similarly, an operator-based approach is used by Elshaafi et al. [24] to predict the trust of compositions. They assume that the trust of composition can be predicted by identifying set of properties (which they call as trustworthiness properties) such as reputation, reliability and security properties. 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. However, in contrast to the other methods(Jaeger et al.'s and Hwang et al.'s), their work includes 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 trustworthiness changes with time, their approach shows the difference better than the other approaches. However, their results use simulations instead of actual services or compositions, and its debatable whether their technique would be accurate for practical applications specially when these properties depends on external factors.

Alagar et al. [25] propose a formal approach for trustworthy composition considering safety, security and time factors. Their composition model requires the participating components to be fully trustworthy. However, as it is common to use untrustworthy components in a composition (specially when third party libraries are required), it is important that the composition model be more flexible to include the use of untrustworthy components in the composition. In contrast, our model proposes composition operators that can be applied on components with different levels of trustworthiness.

Buford et al. [26] propose the idea of composition trust binding (*i.e.*, a policy for composition) to integrate trusted systems among pervasive devices. However, such a model restricts the composition only among a predefined set of services, which has a lower applicability in today's pervasive computing world because of the dynamic nature of the services. Our model assumes the presence of autonomous evaluations of individual services, and derives the trustworthiness of the composed system at the integration time based on various composition operators.

Sherchan et al. [27] present a trust ontology that identifies different types of trust (including composite trust and propagated trust) and their relationships in a service-oriented systems. However, they do not provide a concrete model for assessing the composite trust, which is the main intention of our work.

Mclean el al. [28] and others [29, 30] have proposed formal models for the composition of information flow security properties. Their work shows that to preserve security properties in a composition, all the participating services should at least hold the separability property. In our work, we extend these models to evaluate trustworthiness of the composition of security properties.

Charpentier et al. [31, 32] present a model for reasoning out a composition of universal and existential composition operators. In our work, we use these composition operators to find the corresponding trust operators in evaluating trustworthiness of the composition.

All the above mentioned models do not consider context dependencies to QoS/ trust values, and address the problem of assessing trust associated with QoS properties. Therefore, our work contributes to the state of the art of trusted distributed systems research by presenting a formal model to predict trust of distributed systems.

2.1.2 Machine Learning based QoS/Trust Predictions

Hang et al. [13] have used a mixture of beta distribution to represent trust of services/ compositions. They evaluate the trust of participating services from the trust of compositions by considering dependencies of the composition with its participating services. Those dependencies are represented using a Bayesian network. The beta distribution keeps track whether the service is trustworthy or not as a binary variable. Mehdi et al. [33] extend their approach to use multinomial variable instead of a binary variable (which keeps track of different trustworthiness levels) and a mixture of Dirichlet distribution to derive the service trustworthiness. Trustworthiness of the participating services are derived using Expectation-Maximization algorithm. The main problem with these approaches is that they validate their approach using a syntactic dataset, which may not match with the real life situation. In addition, these approaches do not consider evaluating trust associated with each QoS parameter (most of which are continuous variables) of the system and do not consider the effect of the context on the QoS in their evaluations.

There are other works that use machine-learning techniques to predict the QoS of compositions. A model provided by Eskenazi et al. [34] uses linear regression to predict the performance of software systems that are composed of components. This requires the system developers to identify the 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 the importance of each signature types on 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 manual intervention in identifying signature types (which may not be shared among more than one service) and mapping them to adequate numerical values. This work also does not consider the context of the services in the performance predictions.

In addition to the approaches mentioned earlier [13, 33, 34] (approaches that learn parameters from existing systems, and use the trained models in predicting future systems), there are other works that focus only on extracting performance parameters form existing systems to identify the performance anti-pattern of the system (and possibly improve the performance in next versions). For example, Brosig et al. [12] have proposed a probabilistic approach to capture the data-control flow dependencies to extract the architectural level performance model of a composed system. Similarly, Krogmann et al. [35] use data from reverse engineering of byte-codes to capture the data-control dependencies. However, they do not capture the context-QoS dependencies of existing systems, which we think is an important factor in predicting the QoS of any compositions.

They are existing works that also study context-QoS dependencies of services. For example, Silic et al. [36] present a model that predict reliability of services based on three types of context parameters, i.e, user-specific parameters such as user location, and user profiles, service-specific parameters such as computational complexity, and system resources, and environment-specific parameters such as service provider load, and network performance. Their technique consists of clustering context parameters based on the reliability levels obtained from the execution traces of the services, and predict the reliability of future service based on the cluster of each context parameter. However, their technique will not be much effective in case studies that have continuous QoS parameters (such as response time) and continuous context parameters (such as the resolution of a camera).

Mabrouk et al. [37] presents 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 has changed due to the context change). However, their focus is not to capture the context-QoS dependences of individual services or predict QoS/ trust of service compositions. There are work [38, 39] that consider the associative between services and their effect to QoS in selecting services for a composition. However, their approach cannot be extended to find dependencies from other context parameters (such as physical,input/configuration and execution contexts) to QoS of services and compositions.

Other recent efforts on trust evaluation criteria [40–43] consider only external evidences including ratings and reviews. However, this requires the composed system to be available for use. Therefore, these models are not capable of predicting trust of the composed system in early phases of its lifecycle. In contrast, our approach considers internal evidences as well as external evidences of related systems and is capable of predicting trust values of a system before composing it.

2.2 Optimum Service Selection Problem

There are many efforts that study efficient algorithms to the optimum service selection problem while considering the associativity context of services. Mabrouk et al. [37] present a QoS-aware selection algorithm that re-evaluate the optimization problem and select the best set of services for a composed system with changing QoS values. However, they do not propose a specific model to evaluate context-QoS dependencies and focus on presenting a fast optimization algorithm assuming QoS values are known at any specific point in time. Similarly there are studies [44–46] that focus on fast optimization algorithms to support rapid changes in context. In contrast, our approach studies fast evaluation of context-QoS dependencies along with fast optimization algorithms for the optimum service selection problem with dynamic context.

The model provided by Guo et al. [38] shows that selecting correlated services for a composition would improve the QoS of the composed system. However, their proposed model uses a deterministic algorithm that will not scale with the increase of service groups and the candidate services. Similarly, the model provided by Barakat et al. [39] uses correlations between services to reduce the search space of candidate services while selecting the best subset of services for a particular functional requirement. However, they use greedy approach to check for feasible constraints, hence, it can leads to optimum, and non-feasible solutions. Both these approaches ([38,39]) 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).

There have been many studies [47–50] about optimizing the resource allocations when deploying composed systems on cloud federations. Their works mainly focus on building optimization models that can match available resources (such as CPU, bandwidth, memory) to services for QoS optimization of individual services (specially to minimize the deployment cost of each service). However, these model do not focus on optimizing the composite QoS and trust by selecting the optimum set of services when multiple services with different QoS, but with same functionalities are available. The model proposed by Rekik et al. [47] consider the reconfigurability of services in solving the optimal cloud deployment problem. In contrast, our approach consider many context aspects including the configuration context. Additionally, Our model can be used to optimize the composed QoS and trust of systems when there are multiple competing services available.

There are many theoretical approaches ([14–16]) 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 considers the dependencies between the context and the Qos/trust of services and composed systems.

2.3 Dynamic Adaptations of Composed Systems

The existing work on software adaptation can be divided mainly in to two sub-categories as surveyed by Kell et al. [51]. In the first sub-category, the adaptation is done for the correctness of functionality and in the second sub-category, the adaptation done for the satisfaction of non-functional attributes such as performance, reliability and other QoS. They have indicated that the adaptation done for non-functional attributes is a comparatively harder problem. Since in our work, we are tackling on the problem of adaptation with regard to QoS and trust, below we discuss the previous works in the second sub-category and compare them to our work.

Weyns et al. [52] have surveyed existing formal methods for QoS-aware self adaptive systems and presented statistics on the trends of modeling techniques, adaptation concerns and types of software systems that researchers have been focusing in recent years. Their survey does not report any research with Bayesian networks or probabilistic graphical models as modeling techniques and the trust as an adaptation concern. However, service-based systems are reported as common types of software systems these formal methods have been applied to. Therefore, we think that our model, which uses a Bayesian network as a modeling technique, and trust as an adaptation concern, brings novelty to the self adaptive service-based system research domain.

Cardellini et al. [53] present a QoS-driven adaptation framework specially designed for SOA systems. It has the capability to act as a service broker in the service selection and the interaction pattern selection to adapt composed systems. In contrast, our framework is capable of reacting to context changes in individual service and composed systems, and where possible to neutralize the effects of these changes. In addition to possibly acting as a service broker, our framework feeds input data about the QoS predictions and context evaluations to adaptation services. Their framework focuses only on the satisfaction of average value of QoS. In contrast, our model operates on QoS distributions and the satisfaction of the trust in addition to the satisfaction of the QoS.

Similar to Cardellini et al., Calinescu et al. [54], and Mabrouk et al. [37] model the QoS-based adaptation as an optimization process and use linear programming or heuristic techniques to solve the model equations. Their work is based on the argument that QoS attributes may compete with each other (i.e. improving one attribute may worsen some other attributes [55]), therefore, the optimization technique should consider optimization of attributes simultaneously while satisfying the constraints of individual attributes. In contrast, our model consider the competitive or supportive nature of the QoS attributes implicitly by representing them as nodes in a Bayesian network that connected through paths with either negative (for competitive) or positive (for supportive) parameters.

Villegas et al. [56] present a high-level reference model that uses a self adaptation mechanism to ensure the QoS and Service level agreement are met in dynamic environment. Their approach includes, continuous monitoring of the dynamic context, and feedback loops that automate the dynamics of the system similar to how control theory applications work. However, they have not presented particular concrete techniques that can be used in such feedback loops. We have mainly followed a similar high-level approach, however, in contrast, we present concrete techniques to practically realize the feedback loop. Villegas et al. have decoupled the feedback loop to the sensor process and the effector process. We use the same structure (referred as 'detection' process and 'trigger' process) in the feedback loop of our adaptation model.

Lin et al. [57] propose a framework for adaptive routing for software defined network using a reinforcement learning approach. This learning framework, which had favors the adaptation actions that has improved the network performance in the past for a similar state of the network. In contrast, we use Monte Carlo methods [58] on Bayesian networks to infer the adaption actions. However, we also use caching techniques to remember the actions that had improved the QoS on past for similar contexts and apply them as initial seeds to the Monte Carlo methods.

The adaptation techniques are triggered as a reaction to the changes in the system environment (which we refer to as the context). Most researchers have emphasized the importance of the context in software adaptation process [53, 56]. However, to our knowledge, none of the existing adaptation frameworks explicitly model the dependencies of context to functional and non functional attributes. In contrast, in our model, we consider the 'context' as a first-class citizen and represent it along with the functional and non-functional attributes. With this approach, we are able to build a general model for software adaptation that can continually monitor context changes, detect the QoS and trust violations caused by these changes, and react to recover the QoS and trust in real-time.

In this chapter, we have presented related works in three categories and discussed their drawbacks comparing with our approaches. In summary, we found the following limitations of the prevalent approaches:

- No explicit incorporation of the context in the QoS and trust evaluation during the design phase and the execution phase of system life-cycle.
- No evaluation of trust associated with various QoS attributes that have continuous values for compositions.

In our work, we address these limitations by proposing models that consider context-awareness and trust evaluation as primary objectives.

3 CONTEXT INDEPENDENT QOS AND TRUST PREDICTION FRAMEWORK

The QoS and Trust prediction framework proposed in this chapter is created with the assumption that the QoS and trust of a composition solely depends on the QoS and trust of individual services and their interaction patterns. The prevalent frameworks by Jaeger et al. [22] and Hwang et al. [23] use the same assumption in predicting QoS of compositions. In the work presented in this chapter, we extend their framework to predict trust associated with each QoS of the composition. The content in this chapter is an extension of our previous publications [17, 18].

The proposed context independent QoS and Trust prediction framework consists of two prediction models, 'BDUTrust' and 'RegressionTrust'. In Section 3.1, we introduce a basic independent trust prediction model (referred as 'BDUTrust') in detail. In Section 3.2, we introduce a regression analysis based model (referred as 'RegressionTrust') that uses evidences from existing composed systems to evaluate trust of future systems based on the BDUTrust model. Finally, in Section 3.3, we discuss the drawbacks of the context independent QoS and trust prediction framework, and the improvements we performed to overcome these drawbacks on the novel context dependent QoS and trust prediction framework discussed in Chapter 4.

3.1 The Compositional Trust Model – BDUTrust

3.1.1 Interaction Patterns of Properties

Trust of a composed system can be evaluated by aggregating the trust of functional and quality-of-service (QoS) properties (*e.g.*, response time, availability, and confidentiality) that the system is expected to hold. A single property of a system can either be a boolean assertion (*e.g.*, confidentiality) or a key-value pair (*e.g.*, response time). The trust value of a single property can be derived based on trust values for the corresponding property from each of the individual service involved in the composition. The composition of the individual service properties to create the systemic value, however, depends on the nature of the property and the interaction pattern (*i.e.*, how the services interact with each other) of the software services in the composition.

In this Chapter, we focus on the following QoS properties of software systems: *response time* (*i.e.*, the length of time it take a client to send a request to a server, and receive a response for the originating request; *availability* (*i.e.*, the percentage of the duration that the service is alive since initial deployment); *authentication* (*i.e.*, confirmation/ rejection of an external entity's identify that is trying to access a secure part of the system; *authorization* (*i.e.*, the ability to access a resource in the environment); *confidentiality* (*i.e.*, the ability to ensure that information does not reach parties that are not privy to the information); and *integrity* (*i.e.*, the measure of completeness and correctness of the actions and data). It is worth noting that our approach is not limited to these QoS properties. As discussed in Section 3.1.4, the approach can be applied to other QoS properties by identifying the corresponding composition operators for interaction patterns of the system.

As for the common interaction patterns, we use the patterns originally defined by Hwang et al. [23]:

- **Sequence** – Sequence is when the output of a software service becomes the input of another software service (Figure 3.1). For example, a firewall service is the first service in sequence that authenticates requests to access services behind (or after) the firewall service.

- **Split/Join** – Split/join is when many services shares the input from one preceding service and emit the output to one common, succeeding service (Figure 3.2).

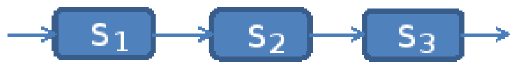


Figure 3.1.: Sequence interaction pattern.

For example, two independent calculations of mathematic equation can be divided into two parallel services, and join them to do the combined calculations.

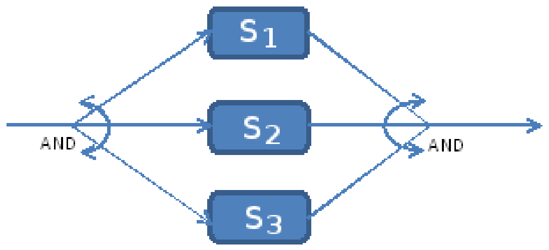


Figure 3.2.: Parallel split/join interaction pattern.

- **Exclusive Choice** – Exclusive choice is when one service from a set of services non-deterministically selected and invoked (Figure 3.3). For example, a load balancing service uses exclusive choice when selecting a service from a set of nearly identical services to obtain a higher availability of the composed system.

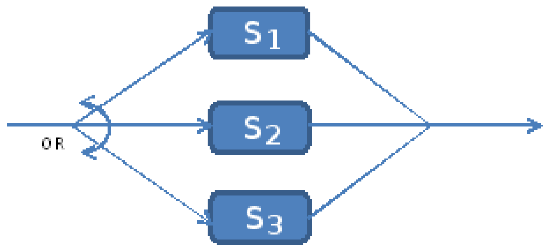


Figure 3.3.: Exclusive choice interaction pattern.

- **Discriminator** – Discriminator is when a service is selected from a set of services that provide the same functional behavior, but different QoS properties

(Figure 3.4). This pattern is different from the exclusive choice pattern above in that here the selection is done based on deterministic criteria. For example, selecting the weather software service that provides the best response time from all the available weather software services.

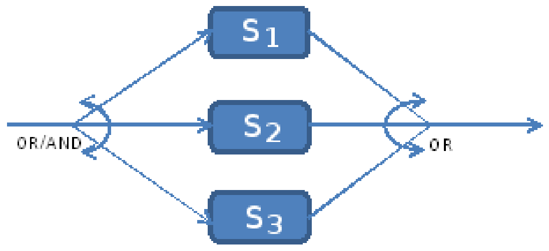


Figure 3.4.: Discriminator interaction pattern.

- **Loop** – Loop is when an output of a service is a feedback as the input of the same service based on some condition (Figure 3.5). For example, selecting a weather service from a set of services on the first iteration, and using the result of the current iteration to select the same, or a different weather service on subsequent iterations.

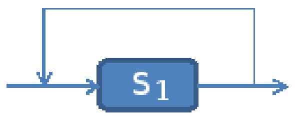


Figure 3.5.: Loop interaction pattern.

- **Gateway** – Gateway is a when a service resides between two other services, and orchestrates communication between the two other services (Figure 3.6). For example, if there are two services that provide heterogeneous interfaces (*e.g.*, CORBA and SOAP), then the gateway is used to enable the communication between either service.

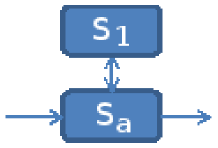


Figure 3.6.: Gateway interaction pattern.

3.1.2 Trust Operators and Representations

For the context independent trust prediction framework, we formally define trust of a service as $T_S = (B, D, U)$ where B is belief that the service conforms to its specification, D is disbelief that the service conforms to its specification, and U is uncertainty that the evidences are incomplete or unavailable to decide the conformance of a service to its specification. We call this the BDU-model and is based on the subjective-logic trust model proposed by Josang et al. [3]. We selected this model as the foundation of our trust compositional model because it captures the uncertainty aspect of the trust, and provides formal operators to evaluate and compose subjective trust values.

Josang et al. also proposes a set of operators on multiple subjective-logic expressions that includes traditional logic operators (*e.g.*, conjunction, disjunction, and negation) and operators specific to the belief theory (*e.g.*, consensus and recommendation). For completeness, we include below the definition of operators originally defined by Josang et al. [3] that are used throughout the remainder of this Chapter. In the list below, the term "agent" is used to represent stakeholders in trust relationships. An agent's A trust of a proposition or another agent's recommendation p is denoted as $T_p^A \equiv < b_p^A, d_p^A, u_p^A >$. When only one agent evaluates trust, trust is represented as $T_p \equiv < b_p, d_p, u_p >$.

- **Conjunction ($\wedge_{B,D,U}$).** Given T_p and T_q, then

$$T_{p \wedge q} \equiv < b_{p \wedge q}, d_{p \wedge q}, u_{p \wedge q} >$$

where,

$$\begin{aligned} b_{p\wedge q} &= b_p \cdot b_q \\ d_{p\wedge q} &= d_p + d_q - d_p \cdot d_q \\ u_{p\wedge q} &= b_p \cdot u_q + u_p \cdot b_q + u_p \cdot u_q \end{aligned}$$

- **Disjunction** $(\vee_{B,D,U})$**.** Given T_p and T_q, then

$$T_{p\vee q} \equiv < b_{p\vee q}, d_{p\vee q}, u_{p\vee q} >$$

where,

$$\begin{aligned} b_{p\vee q} &= b_p + b_q - b_p \cdot b_q \\ d_{p\vee q} &= d_p \cdot d_q \\ u_{p\vee q} &= d_p \cdot u_q + u_p \cdot d_q + u_p \cdot u_q \end{aligned}$$

- **Recommendation** $(\otimes_{B,D,U})$**.** Given agents A and B, T_B^A, and T_p^B, then

$$T_p^{AB} \equiv < b_p^{AB}, d_p^{AB}, u_p^{AB} >$$

where,

$$\begin{aligned} b_p^{AB} &= b_p^A \cdot b_p^B \\ d_p^{AB} &= b_p^A \cdot d_p^B \\ u_p^{AB} &= d_B^A + u_B^A + b_B^A \cdot u_p^B \end{aligned}$$

3.1.3 Trust of Properties in a Composition

As discussed earlier, we define trust of a service to be the degree of confidence that service conforms to its specification. Although there are broader definitions of trust [59], we use this definition, as our focus in this paper is to evaluate trust during the construction of distributed systems. Following this definition, the trust of a service is evaluated by measuring the degree of the compliance. Furthermore, the trust of a composition depends on the properties of its individual services and their interconnection patterns.

It is therefore important to identify the types of relationships that exist between properties of the composed system and the individual services. In the BDU-model, we identify such relationships and the corresponding trust operators for predicting the trust of the composition from the trust of individual services. More specifically, the trust about a property P of a service S defined as:

$$T(P_S) \equiv < B(P_S), D(P_S), U(P_S) > \tag{3.1}$$

If the service S is composed of individual services $S_1, S_2, \ldots S_n$ with an interaction pattern I, then property P of the service S is evaluated as a function of P for the individual services. This function is called the composition operator OP_{P_S} and is defined as:

$$OP_{P_S} : \{P_{s_1}, P_{s_2}, ..., P_{s_n}, I\} \rightarrow P_S \tag{3.2}$$

As shown in Equation 3.2, OP_{P_S} depends on P and the interaction pattern (I) between individual services.

The trust of a property P in the service S depends on OP_{P_S} and is defined by the trust composition operator $OP_{T(P_S)}$ where

$$OP_{T(P_S)} : \{OP_{P_S}, T(P_{s_1}), \ldots, T(P_{s_n})\} \rightarrow T(P_S) \tag{3.3}$$

3.1.4 Trust Composition Operators

Existential operator. If $OP_{P_S} = \exists$, then the composed system will have property P, if there exists a service S_i that satisfies property P. Likewise, the corresponding trust operator for this composition is the disjunction operator. For example, availability (in the discriminator pattern) illustrates usage of the existential operator because when one service has the availability property, then the composed system also has the availability property.

Universal operator. If $OP_{P_S} = \forall$, then the composed system will have property P, if all its services hold property P. Likewise, the corresponding trust operator for this composition is the conjunction operator. For example, confidentiality (in the sequence pattern) illustrates usage of the universal operator because all services in the composed system must have the confidentiality property for the composed system to have the confidentiality property.

Min/Max operator. If $OP_{P_S} = min(x)$ or $OP_{P_S} = max(x)$, then the value of property P of the composed system is the minimum or maximum value, respectively, of property P out of all services used in the composition. For example, response time (in the discriminator pattern) illustrates usage of the minimum operator because the response time of the composed system is equal to the minimum response time of the participating services. Similarly, response time (in the split/join pattern) illustrates usage of the maximum operator because the response time of the composed system is equal to the maximum response time of the participating services.

If the property P is a boolean assertion, then the minimum (or maximum) operator is equivalent to the universal operator. In this situation, $T(P_S) = \wedge_{(B,D,U)}$ (or $T(P_S) = \vee_{(B,D,U)}$). If the property is not a boolean assertion (*i.e.*, a key-value pair such as response time), then $T(P_S)$ must rely on distributions of the operands.

For example, response time is a key-value pair property and example values can be described using a distribution of values with mean 35ms and 4ms standard deviation. Assuming that two separate services (*i.e.*, S_1 and S_2) exhibit the response time distributions above, then the trust value for P of the composed system can be calculated by using probability theory and the corresponding minimum (or maximum) operator to join each service's probability distribution [60]. The resulting joined probability distribution is then used to calculate the trust of P for the composed system. Furthermore, we can approximate that the min/max operator is equivalent to the universal operator for non-binary properties as well.

Addition operator. If $OP_{P_S} = \sum$, then the value of property P for the composed service system is the sum of the property P for all services used in the composition. For example, response time (in the sequence pattern) illustrates the usage of the addition operator because the response time of a composed system is equal to the sum of the response times of the participating services (assuming that each services is running on its own host and there is no competition for CPU resources on the host).

If P is a binary assertion, then the addition operator is equivalent to the existential operator (see above). If P is not a boolean assertion, then the trust value for P of the composed system can be calculated by using probability theory and the corresponding addition operator to join each service's probability distribution [61]. Furthermore, we can approximate that the addition operator is equivalent to the existential operator for non-binary properties as well.

Multiplication operator. If $OP_{P_S} = \prod$, then the value of property P for the composed system is the product of property P for each service used in the composition. For example, availability (in the sequence pattern) illustrates usage of the multiplication operator because the availability of a composed system is equal to the multiplication of the availabilities of the participating services.

If P is a boolean assertion, then the multiplication operator is equivalent to the universal operator (see above) because both produce the same results. If the property is not a boolean assertion,then the trust value for P of the composed system can be calculated by using probability theory and the corresponding multiplication operator to join each service's probability distributions. [60]. Furthermore, we can approximate that the multiplication operator is equivalent to the universal operator for non-binary properties as well.

Multiplier operator. If $OP_{P_S} = multiplier$, then the value of property P for the composed system (P_S) is equal to the multiplication of the property that participate in the composition (P_s) with some constant multiplier n. We denote this as $P_S = n \cdot P_s$. For example, response time (in the loop pattern) illustrates usage of the multiplier operator because the response time of a composed system in loop pattern is equal to the response time of the participating service multiplied by the number of loops.

The application of the multiplier operator is equal to applying the addition operator to a service n times. The trust operator corresponding to the multiplier operator is therefore equivalent to applying trust operator corresponding to the addition operator n times. For example, if the response time of a service is 15ms and is composed in a loop pattern of 10 iterations, then the response time of the composed service is 150 msec (assuming that there is no competition for CPU resources on the host in each iteration). If the Belief component of the trust of the participating service is high, then the trust of the composed system can be obtained by applying the conjunction operator on the on the trust of the participating service n times.

Exponent Operator. If $OP_{P_S} = exponent$, then the value of property P for the composed system (P_S) will be equal to the exponential value of the property that participate in the composition P_s with some constant exponent n. We denote this as $P_S = {P_s}^n$. For example, availability (in the loop pattern) illustrates usage of the exponent operator because the availability of a composed system in loop pattern is equal to exponentiation of the availability of the participating service where the exponent is number of loops.

The application of exponent operator is equal to applying multiplication operator for to a service a constant number of times. The trust operator corresponding to the exponent operator is therefore equivalent to applying the trust operator corresponding to the multiplication operator a constant number of times.

Non-deterministic operator. If $OP_{P_S} = ND$, then the value of property P for the composed system equals the value of P for the non-deterministically selected service. As shown in Equation 3.4, the application of composition operator for a set of services will give an expected value for P equal to the probability that the individual service is selected.

$$P_S = (w_1 \cdot P_{s_1}) + (w_2 \cdot P_{s_2}) + \ldots + (w_n \cdot P_{s_n}) \tag{3.4}$$

For example, availability (in the exclusive choice pattern) illustrates usage of the non-deterministic operator because the availability of a composed system in exclusive choice is equal to the availability of a non-deterministically selected service.

The trust composition operator corresponding to the non-deterministic composition operator is found by applying both the recommendation operator $\otimes$ and disjunction operator $\vee$. As shown in Equation 3.5, the probabilities of individual service S_i being selected is expressed as a subjective logic expression W_i where W_i is the weight that service S_i to be selected considering uncertainty also as a factor.

$$\begin{aligned} T(P_S) &= (W_1 \otimes T(P_{s_1})) \vee (W_2 \otimes T(P_{s_2})) \vee \\ &\quad \ldots \vee (W_n \otimes T(P_{s_n})) \end{aligned} \tag{3.5}$$

Generative operator. If OP_{P_S} is the generative operator, then the composed system will have the value of property P if the condition in the context (denoted as ξ) of the composition is met—even if services do not have property P. For example, the confidentiality property (in the split/join pattern) illustrates usage of the generative operator because by splitting user data flow at different levels (*e.g.*, authorized users and non-authorized users) the control and the data can be made confidential. Here the generative condition ξ is the whether the communications of participating services are separated (without any back-channels) or not. Lastly, the trust composition operator corresponding to the generative operator is equal to the trust that the generative condition ξ for the satified participating services.

Select operator. The select operator is a transformational composition operator that has to be combined with another composition operator. We denote the select operator as $OP_{P_S}[criteria]$ where *criteria* is a condition that a subset of the participating services must satisfy. When the select operator is present, the relevant composition operator is applied only on the participating services that meet the given criteria. For example, authentication (in the discriminator pattern) illustrates the usage of the select operator because the authentication of the composed system is equal to the authentication provided by the participating services with minimum response time. The subset of available services can be selected using the criteria “services having minimum response time”.

When the select operator is applied to property P with a composition operator OP_{P_S}, the trust of P in the composed system is evaluated by applying the trust operator of the related composition operator on the set of services within the composition that hold the criteria. For example, the trust of the authentication of a composed system in the discriminator pattern is calculated by applying the conjunction operator (trust operator corresponding to the universal operator) for the trust of selected services with minimum response times. Lastly, the trust operator is denoted as $OP_{T(P_S)}[criteria]$.

3.1.5 Mapping Composition Operators for Properties and Interaction Patterns

The composition operators and trust operators depend on the interaction patterns between individual services and the of the property under evaluation. For example, evaluating the response time of a system using services that exhibit the sequence interaction pattern requires using of the addition operator. Table 3.1 and Table 3.2 provide an overview of composition operators for common QoS properties with common interaction patterns.

As shown in Table 3.1 and Table 3.2, the universal operator is the composition operator for the confidentiality and integrity QoS property is the irrespective of the observed interaction pattern. This is because Mclean et al. [28] showed that these properties are preserved only when all the participating services have the separability property, which is the ability to ensure that there are no interactions allowed between users at different authorization levels. To preserve authorization and authentication, however, all the participating services need not have the corresponding property. For example, proxy services [62] are used to add authorization and authentication properties to existing systems that do not contain them, and broker authenticators are used when client and servers do not share a trust relationship [63]. In such cases,

Table 3.1.: The composition operators for different QoS properties and each interaction pattern – Part 1.

	Response Time	**Availability**	**Authentication**
Sequence	Addition	Multiplication	Universal
Parallel Split/ Join	Maximum	Multiplication	Universal
Exclusive choice	Nondeterministic	Nondeterministic	Universal
Discriminator (minimum response time)	Minimum	Existential	Universal [minimum response time]
Loop	Multiplier	Exponent	Universal
Gateway	Addition	Multiplication	Universal[gateway service]

Table 3.2.: The composition operators for different QoS properties and each interaction pattern – Part 2.

	Confidentiality	**Integrity**	**Authorization**
Sequence	Universal	Universal	Universal
Parallel Split/ Join	Universal/ Generative	Universal/ Generative	Universal
Exclusive choice	Universal	Universal	Universal
Discriminator (minimum response time)	Universal	Universal	Universal [minimum response time]
Loop	Universal	Universal	Universal
Gateway	Universal	Universal	Universal[gateway service]

the authorization and authentication property values of the composed system depend only on the corresponding property values of the proxy and broker services.

3.1.6 Effect of Environment on Composition Operators

Discussion so far in this chapter assumes that the composition operators depend only on the property under evaluation and the interaction pattern of the composition.

The effect of the execution environment, however, is also an important factor that must be considered when evaluating trust values of the composed system, and its services. For example, if the runtime environment (*e.g.*, the operating system) is not trustworthy, the fully trusted services that execute within the environment will not function as expected. Any trust composition model therefore must be able to integrate trust of the runtime environment.

Building on Equation 3.2, we denote a service S_x running in an environment e as S_x^e, and the property P of the service that runs on the environment e as $P_{s_x^e}$. The composition operator for e is denoted as $OP_{P_S}^e$, as shown in Equation 3.6:

$$OP_{P_S}^e : \{P_{s_1}^e, P_{s_2}^e, \dots, P_{s_n}^e\} \to P_S^e. \tag{3.6}$$

In most situations, we can capture the environment's effect on the composed system as service that participates in the composition. We denote the environment as a service as P_{s_e}. For example, the network that hosts a set of services can be represented as a service, and network delay and network bandwidth as the network service's QoS properties. We can then reuse the environment independent composition operator to evaluate properties of the composition using the following equation:

$$OP_{P_S} : \{P_{s_1}, P_{s_2}, \dots, P_{s_n}, P_{s_e}\} \to P_S^e$$

Similarly, we can extend Equation 3.3 to evaluate the trust of the composition while integrating the as follows:

$$\begin{aligned} OP_{T(P_S)} : \quad & \{OP_{P_S}, T(P_{s_1}), \\ & T(P_{s_2}), ..., T(P_{s_n}), T(P_{s_e})\} \to T(P_S^e) \end{aligned} \tag{3.7}$$

As shown in Equation 3.7, the trust of a property in the composed system is evaluated based on the trust of the property in the participating services and the environment $T(P_{s_e})$. For example, if there are two services connected in a sequence have response times 15 msec each, and the overall network delay of the system is 5 msec, then the response time of the composed system can be calculated by considering the network as another service with the response time of 5 msec. The resulting response time therefore would be 35 msec (assuming that each services is running on its own host and there is no competition for CPU resources on the host).

In the ContextTrust model, introduced in Chapter 4, we consider the environment as an important aspect of evaluating QoS and trust of services and compositions. However, in the subjective logic-based compositional trust prediction we consider the environment only when evaluating the network delay and its effect on the response time of the composition.

3.1.7 Experimental Evaluation of the Compositional Trust model

This section discusses how we applied the trust composition operators introduced in preceding sections to an indoor object tracking system [64] to empirically validate the proposed trust composition model.

3.1.7.1 Overview of the Indoor Tracking System

The indoor object tracking system is a system that tracks the position of objects using a set of heterogeneous sensors. It is also a SOA that is composed from the following services:

- **Sensor services.** Sensor services represent physical sensors that identify the location of the objects in the environment, such as a laboratory or office. Examples of sensor services include, but is not limited to: camera services, wireless network signal tracking services, and Radio Frequency Identification (RFID) services.

- **Fusion service.** The fusion service fuses the outputs of multiple heterogeneous sensor services and outputs an accurate position of the tracked object based on the aggregation of information obtained from each sensor service.
- **Discovery service.** The discovery service selects the most appropriate sensor and fusion services that matches the user's functional and QoS requirements. In this case study, the discovery service selects the two sensor services that most accurately tracks objects while maintaining a low response time.

For the indoor tracking system, we are concerned with evaluating the following QoS properties:

- **Response time (RT).** Response time is how long it takes to system to return the position of the tracked object. Because the object's position can change over time, it is important that response time be low as low as possible to ensure the obtained position is up-to-date.
- **Error of tracked position.** Error of tracked position is the distance between the actual object position and the position provided by the tracking system. The error should be as low as possible to increase the indoor object tracking system's accuracy.

Lastly, while evaluating these QoS properties, we assume the environment (*i.e.*, the network, the devices, and operating system) is a fully trusted service. This means that we exclude the environment in our compositional trust model, but does not mean we cannot include the environment if necessary. For example, if the environment's network has a lot of jitter (*i.e.*, the specified response time of the network is not trustworthy), then the network's trust should also be included when evaluating trust of the composed system.

We selected this case study, because not only it contains distributed heterogeneous components, but also it has strict QoS requirements in the form of response time and accuracy.

3.1.7.2 Experimental Setup

Figure 3.7 shows the experimental setup of the indoor tracking system. As shown in this figure, three camera sensors track an object that is migrated between ten different predefined positions within the view range of the each camera. As the object is tracked, we measure the response time and error of each individual service and the composed system (*i.e.*, the indoor tracking system). At each position, we measure 50 samples of response time and error of the tracked position. We then calculate the uncertainty of each measured QoS property using the ratio of number of failures to provide the object's location (*e.g.*, not recognizing the object) to the number occurrences the service (individually or included in a composition) is invoked throughout its execution lifetime.

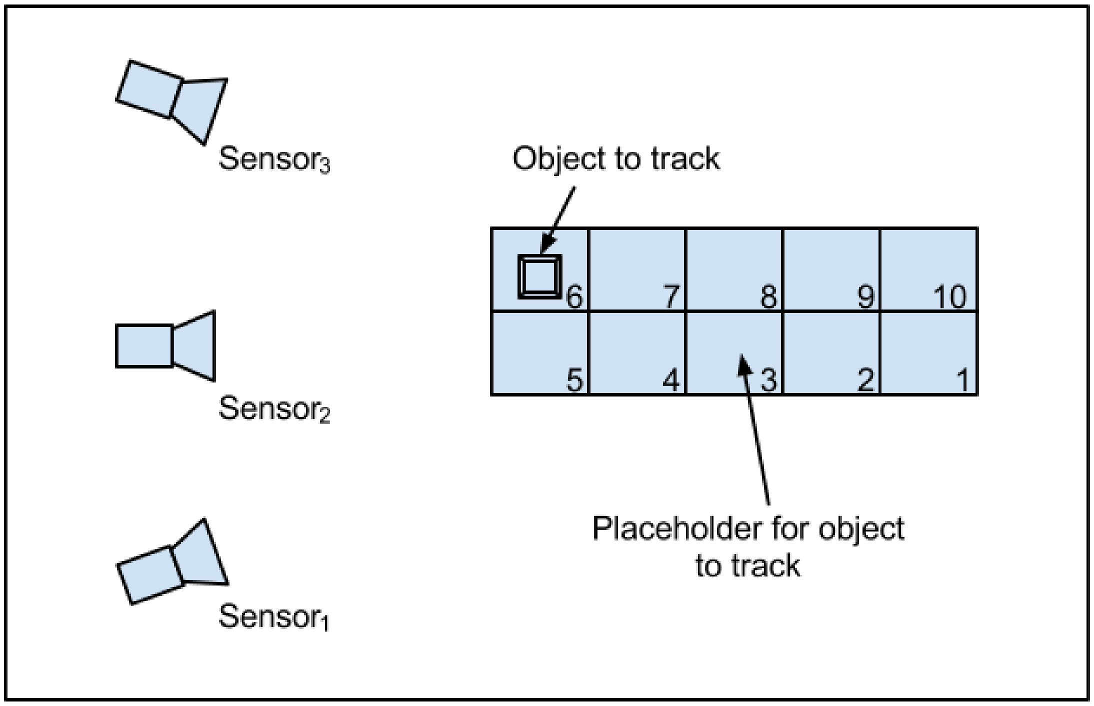


Figure 3.7.: Illustration of the indoor tracking system setup.

Similarly, the belief and disbelief values of the QoS properties are calculated using the ratio of the number of occurrences that the reading adheres to a specified threshold value for the corresponding property to the number of occurrences the reading does not adhere to the threshold value. This calculation is based on past experience, and is considered "seeding" the composition model. For example, the response time of the sensor2 was below 15ms in 304 readings, higher than 15ms in 46 readings, and the sensor was unable to recognize the object in 150 readings. Using the above concept, the trust of the specified response time of the sensor2 (*i.e.*, 15ms) is $< 0.608, 0.092, 0.3 >$. The specifications of QoS properties and trust values based on our past experience with the indoor track system are shown in the Table 3.3 and Table 3.4.

Table 3.3.: Specification of response time and trust of services

Service	**Response time**	**Trust** $< B, D, U >$
Sensor1 (s_1)	15ms	$< 0.59, 0.01, 0.40 >$
Sensor2 (s_2)	15ms	$< 0.608, 0.092, 0.3 >$
Sensor3 (s_3)	15ms	$< 0.328, 0.072, 0.6 >$
Fusion Service (s_4)	4ms	$< 0.616, 0.084, 0.3 >$
Discovery Service (s_0)	20ms	$< 0.623, 0.077, 0.3 >$
Environment (s_5)	3ms	$< 1, 0, 0 >$
Composed system (S)	42ms	$< 0.58, 0.12, 0.3 >$

Table 3.4.: Specification of error and trust of services

Service	**Error**	**Trust** $< B, D, U >$
Sensor1 (s_1)	19cm	$< 0.600, 0, 0.400 >$
Sensor2 (s_2)	19cm	$< 0.60, 0.10, 0.30 >$
Sensor3 (s_3)	19cm	$< 0.398, 0.002, 0.60 >$
Fusion Service (s_4)	Correction ratio: 0.842	$< 0.614, 0.086, 0.3 >$
Discovery Service (s_0)	N/A	N/A
Environment (s_5)	N/A	N/A
Composed system (S)	16cm	$< 0.53, 0.17, 0.3 >$

Lastly, Figure 3.8 shows the complete service interaction diagram of the indoor object tracking system. As shown in this figure, the discovery service selects two of the most appropriate sensors according to accuracy and response time requirements, which are referred as non-deterministically select sensor services. These services then track the object in parallel (referred to as composed sensor service) and provide the tracking result to the fusion service. Lastly, the fusion service outputs the tracked position to the client through the environment service, which is negligible in our experiments.

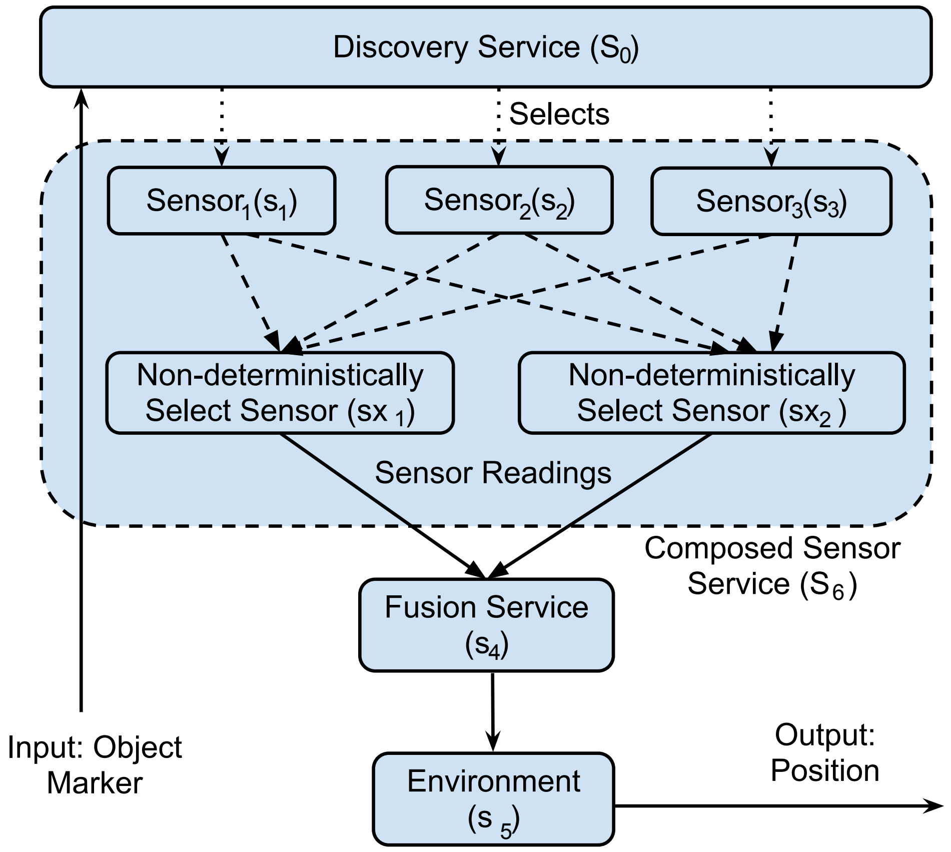


Figure 3.8.: Interaction patterns for the indoor object tracking system.

3.1.7.3 Experimental Results of Compositional Trust model

As mentioned Section 3.1.7.1, response time and error of tracked position are calculated by following a composing pattern of sensor, discovery, and fusion services. The non-deterministically selected sensor services (sx_1, sx_2) are composed of three sensor services (s_1, s_2, s_3) using the exclusive choice pattern as shown in Figure 3.9. When the composition operator and the trust composition operator related to the exclusive choice pattern are applied, the following composed property value and the corresponding trust values are obtained for each property:

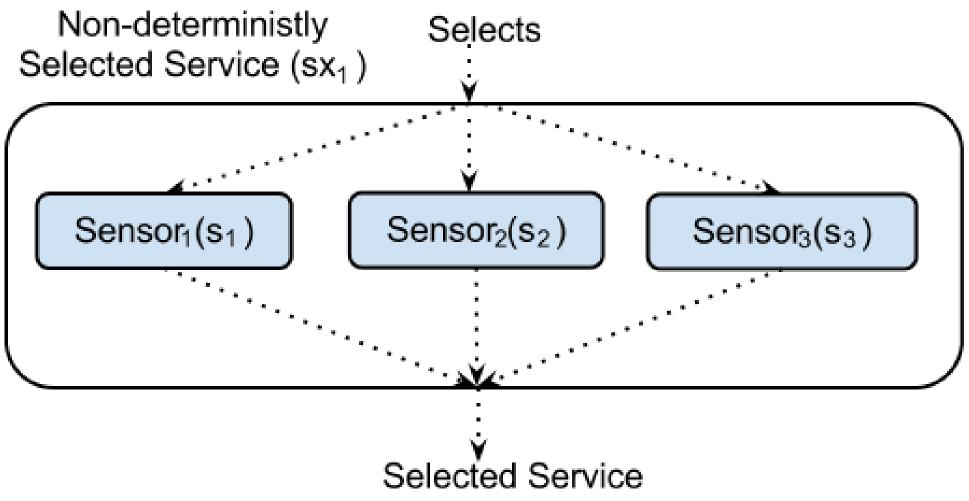


Figure 3.9.: Non-deterministically select sensor services.

Composed response time:

$$\begin{aligned} Rt_{sx_1} &= w_1 \cdot Rt_{s_1} + w_2 \cdot Rt_{s_2} + w_3 \cdot Rt_{s_3} \\ &= \tfrac{1}{3} \cdot 15 + \tfrac{1}{3} \cdot 15 + \tfrac{1}{3} \cdot 15 \\ &= 15ms \end{aligned}$$

Trust of composed response time:

$$\begin{aligned}
T(Rt_{sx_1}) &= W_1 \otimes T(Rt_{s_1}) \vee W_2 \otimes T(Rt_{s_2}) \vee W_3 \otimes T(Rt_{s_3}) \\
&= < 0.34, 0.66, 0 > \otimes < 0.59, 0.01, 0.4 > \\
&\vee < 0.34, 0.66, 0 > \otimes < 0.608, 0.092, 0.3 > \\
&\vee < 0.34, 0.66, 0 > \otimes < 0.328, 0.072, 0.6 > \\
&= < 0.554, 0.123, 0.433 >
\end{aligned}$$

Composed error:

$$\begin{aligned}
Er_{sx_1} &= w_1.Er_{s_1} + w_2.Er_{s_2} + w_3.Er_{s_3} \\
&= \tfrac{1}{3} \cdot 19 + \tfrac{1}{3} \cdot 19 + \tfrac{1}{3} \cdot 19 \\
&= 19cm
\end{aligned}$$

Trust of composed error:

$$\begin{aligned}
T(Er_{sx_1}) &= W_1 \otimes T(Er_{s_1}) \vee W_2 \otimes T(Er_{s_2}) \vee W_3 \otimes T(Er_{s_3}) \\
&= < 0.34, 0.66, 0 > \otimes < 0.6, 0, 0.4 > \\
&\vee < 0.34, 0.66, 0 > \otimes < 0.6, 0.1, 0.3 > \\
&\vee < 0.34, 0.66, 0 > \otimes < 0.398, 0.002, 0.6 > \\
&= < 0.566, 0.001, 0.433 >
\end{aligned}$$

In the equations above, we have assumed each camera service will be selected with equal probability (*i.e.*, $\frac{1}{3}$). Similarly, the property value and the trust value of the second non-deterministically selected sensor service P_{sx_2} can be calculated.

Composed sensor service s_6 is composed of two non-deterministically selected sensor services (*i.e.*, sx_1 and sx_2) using the split/join pattern as shown in Figure 3.10. The properties of the sensor composed service are evaluated as shown in the following equations:

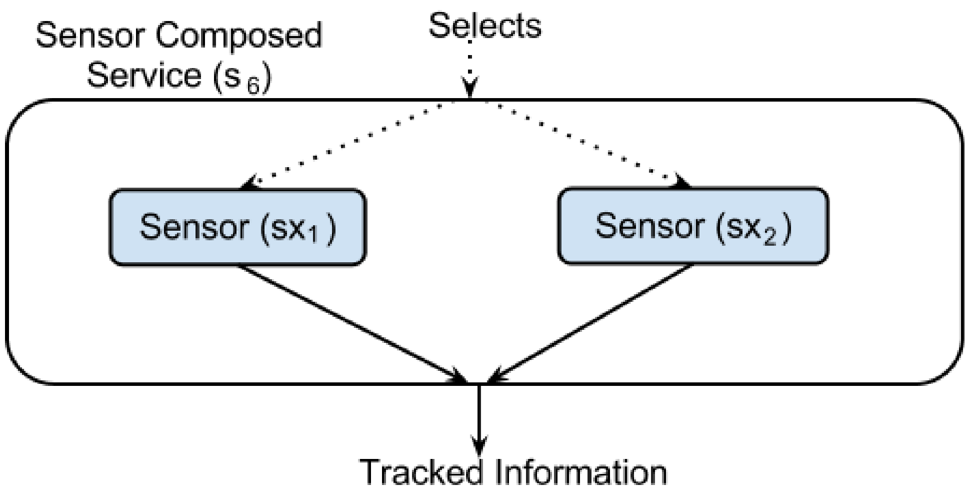


Figure 3.10.: Sensor composed service.

Composed response time:

$$\begin{aligned} Rt_{s_6} &= max(Rt_{sx_1}, Rt_{sx_2}) \\ &= max(15, 15) = 15ms \end{aligned}$$

Trust of composed response time:

$$\begin{aligned} T(Rt_{s_6}) &= T(Rt_{sx_1}) \wedge T(Rt_{sx_2}) \\ &= < 0.554, 0.123, 0.433 > \wedge < 0.554, 0.123, 0.433 > \\ &= < 0.319, 0.014, 0.667 > \end{aligned}$$

Composed error:

$$\begin{aligned} Er_{s_6} &= average(Er_{sx_1}, Er_{sx_2}) \\ &= average(19, 19) \\ &= 19cm \end{aligned}$$

Trust of composed error:

$$\begin{aligned} T(Er_{s_6}) &= T(Er_{sx_1}) \wedge T(Er_{sx_2}) \\ &= <0.566, 0.001, 0.433> \wedge <0.566, 0.001, 0.433> \\ &= <0.57, 0, 0.43> \end{aligned}$$

Finally, the object tracking system S is composed of discovery service s_0, the composed sensor service s_6, fusion service s_4, and the environment s_5 using the sequence pattern as shown in Figure 3.11. The properties of the object tracking system are evaluated as shown in the following equations:

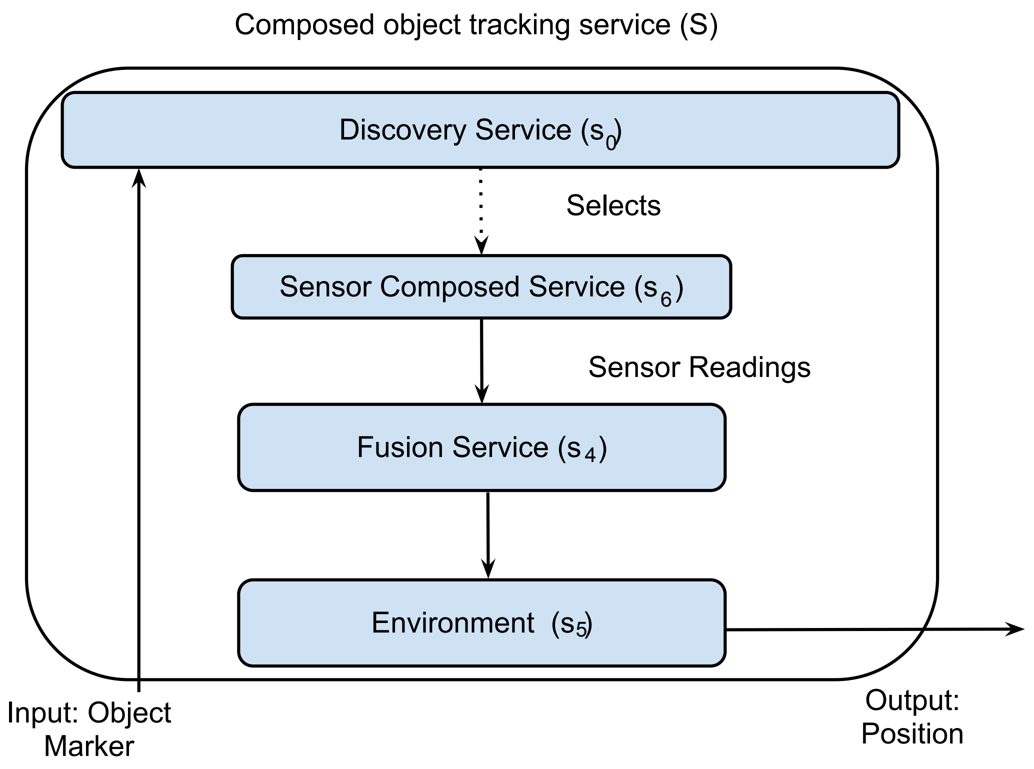


Figure 3.11.: Composed object tracking service.

Composed response time:

$$\begin{aligned} Rt_S &= Rt_{s_0} + Rt_{s_6} + Rt_{s_4} + Rt_{s_5} \\ &= 20 + 15 + 4 + 3 \\ &= 42ms \end{aligned}$$

Trust of composed response time:

$$\begin{aligned} T(Rt_S) &= T(Rt_{s_0}) \wedge T(Rt_{s_6}) \wedge T(Rt_{s_4}) \wedge T(Rt_{s_5}) \\ &= < 0.623, 0.077, 0.3 > \wedge < 0.319, 0.014, 0.667 > \\ &\wedge < 0.616, 0.084, 0.3 > \wedge < 1, 0, 0 > \\ &= < 0.263, 0.000, 0.737 > \end{aligned}$$

Composed error:

$$\begin{aligned} Er_S &= Er_{s_6}.Er_{s_4} \\ &= 19 * 0.842 \\ &= 16cm \end{aligned}$$

Trust of composed error:

$$\begin{aligned} T(Er_S) &= T(Er_{s_6}) \wedge T(Er_{s_4}) \\ &= < 0.57, 0, 0.43 > \wedge < 0.616, 0.084, 0.3 > \\ &= < 0.422, 0.012, 0.566 > \end{aligned}$$

Lastly, the summary of the predicted and the actual trust values of the properties is shown in the table 3.5. As shown in this table, the predicted trust value of the proposed model is not mathematically equal to the actual trust value obtained from executing the composed system empirically. It shows comparatively very high uncertainty associated with the predicted trust value. This is because the calculation process of the predicted trust value accumulates the uncertainty factor with the use of more mathematical operations. However, if the uncertainty is divided between the belief and disbelief with equal weights (indicated as normalized trust values in

Table 3.5), then the predicted and actual trust value become comparable. The predicted trust value therefore conforms to the actual trust value obtained from empirical evaluations while presenting more uncertainty over the predicted results.

Table 3.5.: Predicted and actual values of properties and trust

Trust of Property	**Predicted**	**Actual**
Trust of RT	$< 0.263, 0.0, 0.737 >$	$< 0.58, 0.12, 0.30 >$
Trust of Error	$< 0.422, 0.012, 0.566 >$	$< 0.53, 0.17, 0.30 >$
Normalized Trust of RT	$< 0.632, 0.368, 0 >$	$< 0.73, 0.27, 0 >$
Normalized Trust of Error	$< 0.806, 0.194, 0 >$	$< 0.705, 0.295, 0 >$

3.2 Trust Composition Based on Regression Analysis – RegressionTrust

The experiments in section 3.1.7.3 show that as the complexity of the composition grows, the uncertainty of the prediction also grows. One of the reason for this growth is that the prediction calculation uses a limited number of evidences (namely, the trust of individual services and the service interaction pattern) in evaluating the trust value of the composition. Therefore, in this section, we develop RegressionTrust model as an extension to BDUTrust model with the incorporation of more evidences in evaluating the trust of a composed system.

The motivation behind the proposed approach is to use the evidences from related existing compositional systems in evaluating properties and associated trust of new compositional systems. For example, take a situation that we have a need to evaluate properties and associated trust of a new online gaming system before it is being built. The system is expected to be composed of several services, such as game engine, player management service, social network service, and online scoreboard service. Subsets of these services, such as social network service, and online scoreboard service, can be reused by other existing gaming systems. We can measure the properties and associated trust of these systems and use the measurements in predicting properties and associated trust of the new system.

The proposed approach is to use regression analysis to predict the properties and the associated trust values in new service compositions based on the evidences from related existing compositional systems. Regression analysis allows us to predict the output of a function for a given set of input parameters, when a set of training data is provided. Training data includes set of sample values for input parameters (called as feature space) and the corresponding known outputs of the function. Regression analysis-based machine learning techniques evaluate some parameters (called as model parameters) based on the training data, and use these parameters in evaluating outputs for new inputs.

As our attempt is to predict properties and the associated trust values, we would have four different functions for each property (i.e., one for the value of the property, and three other for the corresponding B, D, and U values). Therefore, we use four different regression model equations with different model parameters to evaluate each of these functions.

The feature space of the regression models (i.e., the set of inputs for the functions) would be candidate services for the compositions. Each candidate service is represented using a binary variable. A binary variable will take the value true, if the corresponding service is included in a particular composition. The impacts of the correlations between the services are represented explicitly in the model by including the multiplication of two binary variables for each binary variable pair into the feature space.

When evaluating value of the property, the proposed model works only when the composition operator for the property is addition, as the linear regression models require the prediction to be linearly dependent on its components. However, most of the composition operators can be transformed into a particular form of addition. For example, multiplication can be transformed to addition by taking the logarithms of the property values; weighted average can be transformed to addition by multiplying weights together with the property values. Hence, linear regression models can be used in evaluating different properties of the composition. Additionally, we assume existing compositions are available with different number of participating services, as then the feature matrix is not left invertible and the training model is not capable of performing inferences.

3.2.1 Machine Learning Model

Let us consider n services, S_1, S_2, ..., S_n. The binary variables (X) corresponding to each of these services are denoted as x_1, x_2, ..., x_n. Then we choose a basis function ($\phi(X)$) to generate a feature space consisting of x_1, x_2, ..., x_n, $x_1.x_2$, $x_1.x_3$, ..., $x_1.x_n$, $x_2.x_3$, ..., $x_{n-1}.x_n$ terms. When we model the problem using the linear regression model, the generated model for the response time of a particular composition (y) can be written as follows (equation 3.8).

$$y = W^T\phi(X) \quad where \quad \phi(X) = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ x_1.x_2 \\ x_1.x_3 \\ \dots \\ x_{n-1}.x_n \end{bmatrix}, \; W = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \\ w_{12} \\ w_{13} \\ \dots \\ w_{(n-1)n} \end{bmatrix} \tag{3.8}$$

Here, the terms w_i correspond to the value of the property of the Service S_i. The terms w_{ij} (where $1 \leq i, j \leq n$) indicate how the correlation between service S_i and service S_j impacts on the overall composed property. If the w_{ij} is greater than zero, the correlation of the services increases the overall property value, whereas if the w_{ij} is less than zero, the overall property value reduces due to the correlation between the services. If the wij is equal or very close to zero, the correlation between services does not impact the composed property value.

Similarly, we define a regression model with different model parameters for each B, D, U value that represents the trust associated with a particular system property. Although each of these B, D, U values is not linearly composable (because conjunction is not a linear operator), here we assume they are approximately linearly composable as an approximate heuristic. This assumption provides our models with the capability to use relevant information about the trust of associated systems in the prediction of the system under consideration. We denote the Belief of a particular property of the composed system to be B_y and the model learning parameters are B. Hence, the corresponding regression model is:

$$B_y = B^T \phi(X) \quad where \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \\ b_{12} \\ b_{13} \\ \dots \\ b_{(n-1)n} \end{bmatrix} \tag{3.9}$$

Similar to the system properties, the term b_i corresponds to the value of the belief of the property of Service S_i and the terms b_{ij} corresponds how the correlation between service S_i and service S_j impacts the belief of the overall composed property. Similarly, we model D and U using the regression model as in following representations:

$$D_y = D^T\phi(X) \quad where \quad D = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_n \\ d_{12} \\ d_{13} \\ \dots \\ d_{(n-1)n} \end{bmatrix} U_y = U^T\phi(X) \quad where \quad U = \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \\ u_{12} \\ u_{13} \\ \dots \\ u_{(n-1)n} \end{bmatrix} \tag{3.10}$$

3.2.2 Experimental Evaluation of RegressionTrust

For the experimental setup, we have used sample services from the travel planning domain; however, services from any other domain, which have multiple services from different providers for the same service category, could have been used as well. The services we used along with the related service categories are shown in the Table 3.6.

In addition to using the publically available services, we have written test services for each category indicated in Table 3.6. These services use an internal database, which is populated with data retrieved from publically available services, and are capable of providing results for a limited number of queries. We have created explicit correlations between some of these services (services that start with a Shared Session prefix are correlated) by keeping user context information in shared session variables to study how our learning model captures these known associations. We used the Bayesian Linear Regression technique to train and predict the models we choose this technique, because the Bayesian model allows us to control the prediction using prior distributions over the learning parameters.

For the learning model in Equation 3.8, we use the actual property values of individual services (i.e., the response times of each services), which are found empirically,

Table 3.6.: Services used in travel planning system

Service Category	**Services**
Direction Services (DR)	Google Direction Service [65] (DR1) MapQuest Direction Service [66] (DR2) Open Maquest Direction Service [67] (DR3) Session-less Test Direction Service (DR4) Shared-Session Test Direction Service (DR5) Shared-Session Test Direction Service2 (DR6)
Traffic Services (TR)	MapQuest Traffic Service [68] (TR1) Session-less Test Traffic Service (TR2) Shared Session Test Traffic Service (TR3)
Hotel Services (HO)	Expedia Hotel Service [69] (HO1) Google Hotel Service [70] (HO2) Yahoo Hotel Service [71] (HO3) Session-less Test Hotel Service (HO4) Shared Session Hotel Service (HO5)
Weather Services (WE)	Government Weather Service [72] (WE1) Ham Weather Service [73] (WE2) Weather2Weather Service [74] (WE3) Weather Channel Weather Service [75] (WE4) World-Weather Weather Service [76] (WE5) Yahoo Weather Service [77] (WE6) Session-less Test Weather Service (WE7) Shared Session Weather Service (WE8)
Car Rental Services (CR)	HotWire Car Rental Service [78] (CR1) Session-less Car Rental Service (CR2) Shared Session Car Rental Service (CR3)

as the prior distributions. We denote the prior distribution of the learning model in Equation 3.8) as W^0 multivariate normal distribution,

$$P(W^0) = N(W^0|M^0, S^0) \quad where \quad M^0 = \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ m_n \\ m_{12} \\ m_{13} \\ \dots \\ m_{(n-1)n} \end{bmatrix} \tag{3.11}$$

Here M^0 is the mean of the prior distribution and $S^0 = [s_{ij}]$ is the associated variance-covariance matrix.

As mentioned above, $N(m_i, s_{ii})$ $(0 \leq i \leq n)$ parameters correspond to the distribution of the property of the individual service S_i. This allows us to keep the predicted training parameters, w_i, close to the value of the property. All the other parameters, i.e., m_{ij}, and s_{ij} are set to zero. That will make sure the w_{ij} parameters to be close to zero, so the model will not keep unnecessary dependencies (i.e., dependencies that are not supported by the training data) between the services.

Similarly, we have used prior distributions for the B, D, U values. Similar to the early learning model (Equation 3.8), the first n parameters are given the empirically evaluated (B, D, U) values (as described in the below paragraph.) for each service, which will make sure that the training parameters will stay close to the individual (B, D, U) values of the services. As the (B, D, U) tuples are evaluated separately, the predicted values could add up to higher than one. Therefore, we normalize the tuple values predicted by the model.

In our experiments, we used 2426 sample compositions from the services shown in the Table 3.6. Two third of them are used to train the model and the rest of them are used as test data to test the error of the model. We assume response times of the compositions behave as Gaussian distributions. We calculated the unbiased estimates of mean (μ) and covariance (σ) of the response times by repeated invocations of composition systems. The Disbelief factor of the Trust value (associated with the response time) is calculated by taking the ratio of the outliers. The outliers are response times in the range $[x < \mu 2\sigma]$ or $[x > \mu + 2\sigma]$. Similarly, the Belief factor is calculated by taking the ratio of the values in the range $[\mu - 2\sigma < x < \mu + \sigma]$. The Disbelief factor will be equal to $1BU$, which corresponds to the ratio of response times in the range $[\mu + \sigma < x < \mu + 2\sigma]$.

Table 3.7.: Comparison of relative absolute errors of the results of models

Composed Property	**Relative absolute error of Regression-Trust**	**Relative absolute error of BDUTrust**
Response Time	0.83895	1.4052
Belief	1.1302	8.1977
Disbelief	0.94567	3.3491
Uncertainty	1.0032	4.4095

Table 3.7 shows the comparison of relative absolute errors associated with the prediction results (for the test data). Here the BUDTrust model uses the addition of the response times of participating services to predict the response time of composed systems. Additionally, the BDUTrust uses the conjunction operation over the trust values of the response times of participating services to predict the trust value (BDU tuple) of composed systems. The Table 3.7 shows that the RegressionTrust model provides better predictions for all the parameters. Tables 3.8, 3.9, and 3.10 show the comparisons between the values provided by the two models along with experimentally evaluated actual values for some selected sample compositions.

The most notable difference between the two models is that the BDUTrust predictions include lesser belief and relatively higher uncertainty about the predicted property. The predictions obtained from the RegressionTrust show lesser uncertainty values, which are comparable with the actual values. We have seen in our previous experiments that as the complexity of the composition increase (i.e., high number of participating services and complex interaction patterns), the uncertainty also in-

Table 3.8.: Predicted values from the BDUTrust model

Composition	**Response Time**	**Trust**
D1+T1+H1+C1+W1	1714	$< 0.55, 0.23, 0.21 >$
D1+T1+H2+C1+W2	1035	$< 0.49, 0.31, 0.19 >$
D1+T1+H3+C1+W2	1118	$< 0.59, 0.19, 0.22 >$
D1+T1+H2+C1+W3	1751	$< 0.52, 0.31, 0.17 >$
D1+T1+H3+C1+W3	1834	$< 0.62, 0.19, 0.19 >$
D1+T1+H3+C1+W4	782	$< 0.59, 0.19, 0.22 >$
D2+T1+H1+C1+W1	1740	$< 0.62, 0.15, 0.24 >$
D2+T1+H3+C1+W1	1150	$< 0.69, 0.10, 0.21 >$
D3+T1+H3+C1+W6	1697	$< 0.59, 0.19, 0.22 >$
D4+T2+H4+C2+W7	1413	$< 0.44, 0.38, 0.18 >$
D5+T2+H4+C2+W7	2748	$< 0.49, 0.27, 0.23 >$
D5+T1+H1+C1+W1	3178	$< 0.55, 0.15, 0.30 >$
D5+T3+H5+C3+W8	5406	$< 0.52, 0.19, 0.28 >$
D6+T2+H4+C2+W7	4157	$< 0.41, 0.49, 0.10 >$
D6+T3+H5+C3+W8	6815	$< 0.43, 0.43, 0.13 >$

Table 3.9.: Predicted values form the RegressionTrust model

Composition	**Response Time**	**Trust**
D1+T1+H1+C1+W1	1332	$< 0.86, 0.11, 0.03 >$
D1+T1+H2+C1+W2	983	$< 0.89, 0.04, 0.06 >$
D1+T1+H3+C1+W2	1043	$< 0.88, 0.07, 0.06 >$
D1+T1+H2+C1+W3	1378	$< 0.89, 0.04, 0.07 >$
D1+T1+H3+C1+W3	1443	$< 0.88, 0.06, 0.07 >$
D1+T1+H3+C1+W4	488	$< 0.87, 0.07, 0.05 >$
D2+T1+H1+C1+W1	1433	$< 0.83, 0.14, 0.03 >$
D2+T1+H3+C1+W1	919	$< 0.83, 0.13, 0.04 >$
D3+T1+H3+C1+W6	1466	$< 0.88, 0.07, 0.05 >$
D4+T2+H4+C2+W7	2466	$< 0.88, 0.05, 0.06 >$
D5+T2+H4+C2+W7	3184	$< 0.85, 0.09, 0.06 >$
D5+T1+H1+C1+W1	2891	$< 0.81, 0.15, 0.03 >$
D5+T3+H5+C3+W8	4895	$< 0.78, 0.17, 0.05 >$
D6+T2+H4+C2+W7	4476	$< 0.79, 0.17, 0.03 >$
D6+T3+H5+C3+W8	6166	$< 0.79, 0.17, 0.04 >$

creases. Similarly, disbelief of the preliminary model can be seen as a worse case prediction, which acts an upper bound for the actual prediction. Additionally, the

Table 3.10.: The actual values from empirical evaluations

Composition	**Response Time**	**Trust**
D1+T1+H1+C1+W1	1358	$< 0.85, 0.15, 0.00 >$
D1+T1+H2+C1+W2	838	$< 0.80, 0.15, 0.05 >$
D1+T1+H3+C1+W2	964	$< 0.85, 0.15, 0.00 >$
D1+T1+H2+C1+W3	1476	$< 0.90, 0.00, 0.10 >$
D1+T1+H3+C1+W3	1541	$< 0.85, 0.10, 0.05 >$
D1+T1+H3+C1+W4	646	$< 0.95, 0.00, 0.05 >$
D2+T1+H1+C1+W1	1680	$< 0.85, 0.10, 0.05 >$
D2+T1+H3+C1+W1	935	$< 0.75, 0.25, 0.00 >$
D3+T1+H3+C1+W6	805	$< 0.80, 0.20, 0.00 >$
D4+T2+H4+C2+W7	2523	$< 0.85, 0.10, 0.05 >$
D5+T2+H4+C2+W7	3403	$< 0.85, 0.10, 0.05 >$
D5+T1+H1+C1+W1	2960	$< 0.85, 0.05, 0.10 >$
D5+T3+H5+C3+W8	4814	$< 0.80, 0.15, 0.05 >$
D6+T2+H4+C2+W7	4934	$< 0.75, 0.25, 0.00 >$
D6+T3+H5+C3+W8	4054	$< 0.70, 0.30, 0.00 >$

results show that when the actual disbelief is low, both the model predicts relatively low disbelief for that composition.

Table 3.11.: Parameters related to service associations

i	j	w_{ij}	b_{ij}	d_{ij}	u_{ij}
D1	H2	-9	-0.150	0.001	-0.014
D1	H3	17	-0.163	0.133	-0.016
D1	W5	1022	-0.094	-0.246	-0.004
D2	T1	-327	-0.241	0.004	-0.003
D1	T3	209	-0.289	0.003	-0.019
D3	T1	11	-0.221	-0.002	-0.018
D5	H5	-1017	-0.014	-0.009	-0.012
D5	T2	53	-0.005	-0.025	-0.011
D6	T2	-438	-0.034	-0.002	-0.009
D6	T3	-2455	-0.006	-0.039	0.004
D6	H5	-1342	-0.024	-0.019	0.007
D6	W8	-1441	-0.041	0.012	-0.009
D6	C3	-1308	-0.044	-0.21	0.024

In order to examine how the associations between services have impacted the results of the RegressionTrust model, we show, in Table 3.11, the parameters of the learning models for some selected service associations. It indicates that when two services are provided by the same company/organization, the association parameter tends to be negative (therefore, reduces the composed response time). Examples are Google Direction Service (D1) with Google Hotel Service (H2), and MapQuest Direction Service (D2) with MapQuest Traffic Service (T1). A reason for this observation can be that both sets of services require the service invocations to provide an authentication key. Therefore, if two related invocations happen, it is possible the providers have optimized the performance by keeping the user context data. This also explains the observation that although the Open-MapQuest Direction Service (D3), and MapQuest Traffic Service (T1) are provided by the same organization, as the Open-MapQuest Direction service does not require a key, the association parameters have not become negative. Thus, the proposed model has successfully captured the explicit associations made using shared session services (they keep the user context in shared session). For example, Services D6, and T3 shares sessions, and thus, the associated parameters have relatively high negative values.

Similar to response time, the association parameters also indicates how the B, D, U values change as the services compose together. However, unlike in response time, experimental results show the services provided by same organization will not make a more trusted composition than services from different organizations. For example, the disbelief has positive values (D1 with H2 and D2 with T1) when the services have been provided by the same organization. The reason for this observation is that the response times of these compositions vary a lot among service invocations (i.e., some invocations perform well better than others).

RegressionTrust performs better in predicting QoS and trust of compositions compared with BDUTrust; however, it has the drawback of needing existing systems that re-uses the same candidate services, which may not hold true for all the scenarios.

Therefore, it is important to study more techniques that would give better predictions of QoS and trust of systems at early stages of the system life-cycle.

3.3 Lessions

Experimental evaluations of the BDUTrust model shows that the context independent trust evaluations yield higher uncertainties in both individual services and composed systems. We identified there are two reasons causing these higher uncertainties.

1. As the QoS evaluations done under a verity of contexts, the corresponding QoS values also have a high variance. In section 3.1.7.2, we evaluate the trust of each sensor services by keeping the object to track in different positions and aggregating the QoS values corresponding to each positions. That has yielded high uncertainty in the trust of these sensors as shown in Table 3.3 and Table 3.4. Similarly, that has caused an higher uncertainty in the trust prediction of the composition.

2. Aggregations of trust values that have contradictory 'Belief' values yields high 'Uncertainty' results. As the experiment discussed in section 3.1.7.2 aggregates the trust values of different contexts, which may have variety of 'Belief' values, the predicated trust value has a higher 'Uncertainty' component. This can be observed from the results shown in Table 3.5.

3. Subjective logic operators such as conjunction, disjunction, and negations can be effectively used only in representing trust of binary QoS properties. In our experiments, we mostly deals with QoS properties like tracked error and response time, which are non-binary properties with continuous values. Therefore, the subjective logic operators only provide approximate aggregated trust values for these QoS properties.

These drawbacks of the context independent trust prediction framework motivated us to develop the next framework with the following improvements.

1. We choose to make the trust predictions context-aware that lead to predictions with lesser variations, and hence lesser uncertainty. From the available evidences, we extract out context-QoS dependencies, and then we use the proposing model to predict the QoS localized to the context that system will run. In the proposing model, we use Bayesian networks to capture such context-QoS dependencies and use sampling based inference techniques to predict the QoS for a particular contexts.

2. We choose to use probability (i.e., B, D representation) as opposed to subjective logic (i.e., B, D, U representation) as the trust evaluation metric. The reasons for this change are:

   (a) Most of the QoS properties have continuous values (in contrast to binary values). With the use of probability representation, we can represent these values as probability distributions. Aggregations of probability distributions with continuous variables are well defined [60, 61]. For non-binary properties, the subjective logic operators only provide approximation aggregated trust values. Therefore, the use of probability distributions for continuous QoS properties provides more accurate predictions.

   (b) Many learning and inferences techniques for Bayesian networks are readily available for probability representations as opposed to subjective logic representations of data. Therefore, we will be able to use these techniques in our model, when we represent trust as a probability.

   (c) Probability distributions such as the Gaussian distribution has parameters such as variance that indicate the uncertainty of the data. Therefore, when we have the trust represented with a probability distribution, we can easily transform the trust representation between subjective logic and probability representation.

With these improvements, we build the context dependent trust prediction framework that is discussed more in detail from Chapter 4.

4 CONTEXT DEPENDENT QOS AND TRUST PREDICTION FRAMEWORK – CONTEXTTRUST

As the context independent QoS and trust prediction framework described in Chapter 3 has resulted predictions with higher uncertainty, we concluded that it is important to consider the context in predicting QoS and trust of compositions. The proposed context dependent QoS and trust prediction framework consists of four models. There are: the basic prediction model (referred as 'ContextTrust') described in detail in this Chapter, optimization model (referred as 'OptimumTrust') discussed in Section 6.1, adaptation model (referred as 'AdaptTrust') discussed in Section 6.2, and trust evaluation model of reusable services (referred as 'ReuseTrust') discussed in Section 6.3. The content in this chapter is an extension of our previous publication [19].

The ContextTrust model uses Bayesian network-based machine learning techniques to capture the context-QoS dependencies in predicting QoS and trust of composed systems. The model mainly consists of four phases.

1. To collect the Context-QoS dependency information of individual services.

2. To collect information about the interaction patterns in the composed system.

3. To derive Bayesian networks for the context-QoS dependency for compositions.

4. To use inference techniques to answer relevant QoS/trust queries.

Each of the above phase is described with the help of a case study involving an Indoor Tracking System [79]. The tracking system is used to track positions of markers placed inside an indoor environment. The tracking system is created by composing three atomic services. The Table 4.1 describes the role of each of the participating services.

Table 4.1.: Services in indoor distributed tracking system

Service	Description
Camera Tracking Service (s_c)	Track the position of a marker relative to a smart phone camera (physical setup of the camera tracking service is shown in Figure 4.1).
WiFi Tracking Service (s_w)	Track the position of a smart phone by triangulation of signal strengths from three WiFi routers (physical setup of the WiFi tracking service is shown in Figure 4.2).
Average Fusion Service (s_f)	Fuse two or more independently predicted positions to derive an accurate position of the marker.

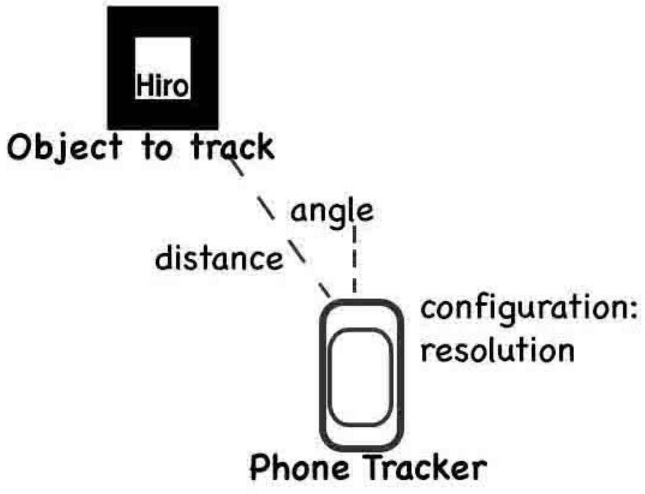


Figure 4.1.: Physical setup of the camera tracking service.

4.1 Phase 1: To Collect the Context-QoS Dependency Information of Services

To predict trust of a future system, the model needs information about the context-QoS dependencies of each participating service. In addition to that, the model keeps track of the context-context, and QoS-QoS dependencies as well. If a

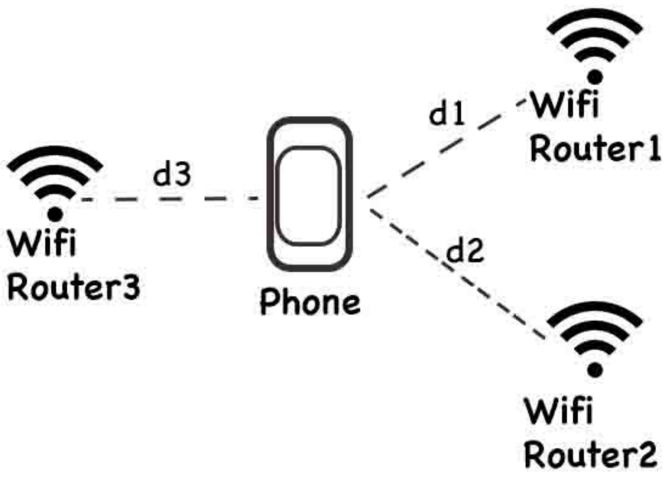


Figure 4.2.: Physical setup of the WiFi tracking service.

service (S) has QoS properties ($Q_S = q_S^1, q_S^2, .., q_S^n$) and associated context properties ($C_S = c_S^1, c_S^2, .., c_S^m$), then there are functions $f_{q_S^x}$, and $f_{c_S^y}$, such that

$$f_{q_S^x} : C_S \rightarrow Q_S \quad for \quad x = 1, 2, .., n \tag{4.1}$$

$$f_{c_S^y} : C_S \rightarrow C_S \quad for \quad y = 1, 2, .., m \tag{4.2}$$

Here $f_{q_S^x}$ indicates the dependencies of the context properties to a QoS property, whereas $f_{c_s^y}$ represents the dependencies between a context property to another context property.

These context-QoS dependencies context-context dependencies, and QoS-QoS dependencies of each service are represented as a Bayesian network [80]. A Bayesian network is a directed graph (G_s) with a set of vertices(V), edges(E) and dependency functions (F) such that,

$$G_s = (V, E, F) \tag{4.3}$$

$$e = (v, u) \quad \forall e \in E \quad where \quad v \in V, u \in V \tag{4.4}$$

$$\begin{gathered}\exists f_u^{bn} \in F, \quad f_u^{bn} : V \rightarrow V \quad such \quad that \quad u = f_u^{bn}(v_1, v_2, ..v_n), \\ \forall u \quad where \quad (v_1, u) \in E, \quad (v_2, u) \in E, \quad .., \quad (v_n, u) \in E\end{gathered} \tag{4.5}$$

If $(v, u) \in E$, then u is called as a child vertex and the v is called as a parent vertex of u. Equation 4.5 indicates the relationship of each child vertex (u) to its parent vertices $(v_1, v_2, .., v_n)$ when there are edges $(v_1, u), (v_2, u), .., (v_n, u)$.

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

$$V = Q_s \cup C_s \tag{4.6}$$

$$\begin{gathered}(c_s^y, q_s^x) \in E \quad if \quad \exists q_s^x = f_{q_s^x}(.., c_s^y, .., c_s^z, ..) \\ for \quad x = 1, 2, .., n, \quad y = 1, 2, .., m, \quad and \quad z = 1, 2, .., n \neq x\end{gathered} \tag{4.7}$$

$$\begin{aligned}(c_s^y, c_s^x) \in E \quad if \quad \exists c_s^x = f_{c_s^x}(.., c_s^y, ..) \quad for \quad x = 1, 2, .., n, \\ y = 1, 2, .., m \quad and \quad x \neq y\end{aligned} \tag{4.8}$$

$$f_u^{bn} = \left\{ \begin{matrix} f_{q_s^x} & if & u = q_s^x & for & x = 1, 2, .., n \\ f_{c_s^y} & if & u = c_s^y & for & y = 1, 2, .., m \end{matrix} \right\} \tag{4.9}$$

The equations 4.6, 4.7, 4.8, and 4.9 can be explained as following.

- Each context property (c_s^x) and QoS property (q_s^x) is mapped into a vertex of the graph (equation 4.6).

- There are edges that are directed either from a context vertex to a QoS vertex or from a context vertex to an another context vertex. (equations 4.7 and 4.8). If there are edges coming towards a particular vertex, we call such a vertex dependent vertex. If a vertex do not have any incoming edges, then such a vertex is called a independent vertex.

- The function (f_u^{bn}) is called as the dependency function. It 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 (equation 4.9).

.

The dependency functions are associated with all the dependent vertices in the Bayesian network. The independent vertices could also be associated with a parametric distribution. In our case studies, we associate uniform distributions or Gaussian distributions with a high variance for the independent vertices.

The advantage of using Bayesian network is that it has a graphical representation that can be easily interpreted by Humans in addition to the software applications. For example, for a camera tracking service (S_c), the QoS properties (Q_{S_c}) are response time (rt), tracked error (e). Associated context parameters (C_{S_c}) are distance to the object (d), angle to the object from the direction perpendicular to the camera face(a), resolution width (w), and resolution height (h). The Bayesian network corresponding to the context-QoS dependencies of camera tracking service is shown in the Figure 4.3. There the d, a, rw, and rh are independent vertices, and ex, ey, e, and rt are dependent vertices.

The representation of a service context-QoS, context-context, and QoS-QoS dependencies in a Bayesian network requires 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)

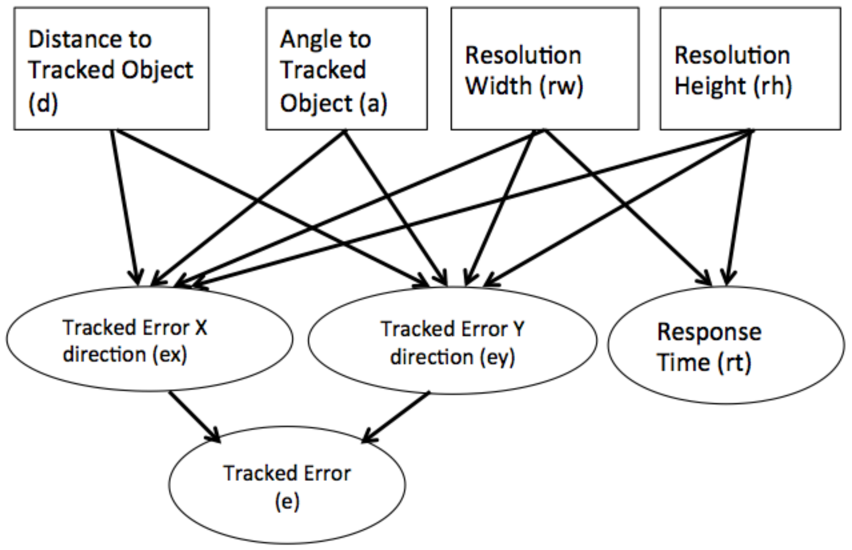

Distance to Tracked Object (d)
Angle to Tracked Object (a)
Resolution Width (rw)
Resolution Height (rh)
Tracked Error X direction (ex)
Tracked Error Y direction (ey)
Response Time (rt)
Tracked Error (e)

Figure 4.3.: Bayesian network of the camera tracking service.

4.1.1 To Evaluate the Structure of the Bayesian Network (Structure Learning)

The structure of the Bayesian network consists of its vertices (V) and the edges (E). This can be evaluated either manually, automatically or semi-automatically.

Manual evaluation have to be done with the help of a domain expert. The expert would first list the important QoS properties of a particular service in the domain, and the context properties that would affect the QoS properties. Those properties would form the vertices of the Bayesian network. Then the expert would indicate the context-QoS, context-context, and QoS-QoS dependencies based on his experience and knowledge about the domain. These connections form the edges of the Bayesian network.

The automatic evaluation requires enough data to learn the structure of the network using machine learning methods. To identify the important context and QoS properties feature selection techniques such as Principle Component Analysis [81], or Singular Value Decomposition [82] can be used. These techniques select the vertices of the Bayesian network. To identify the edges of the network, machine learning algorithms based on likelihood scores [80] can be used.

However, even in the automatic evaluation of the structure, there are many threshold parameters that should be decided by the human experts. Therefore, the structure of the Bayesian network always have to be either evaluated or validated by a domain expert.

4.1.2 To Evaluate the Dependency Functions (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 experts to perform parameter tuning), than the manual techniques. However, as most of the interested QoS properties 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 three of such techniques.

- Using probability tables after discreting the data
- Regularized least squares regression
- Bayesian linear regression with sampling

All these technique requires data points that are obtained from execution traces. Each data point should include the instance values of interested QoS properties and context properties. If the execution traces of individual services are not available, we can not use these techniques directly. In such situations, if the execution traces of composed systems that has reused these services are available, a model called 'ReuseTrust' can be used to learn the parameters of the Bayesian networks of these service. We discuss the 'ReuseTrust' model in detail in Section 6.3.

4.1.2.1 Using probability tables after descreting the data

As there are many inferences techniques such as sum-product algorithm, max-sum algorithm [83] for Bayesian networks that can be easily applied with discreet data, it is very common to first descretize the continuous data to a discrete format [84] and then use the discreet inferences techniques. If there is a continuous property (a_c) with the range (s_{a_c}, e_{a_c}), then it is transformed in to a discreet property (a_d) with the range $(0, n-1)$ by applying the function (g_{cd}). This transformation is equivalent to partitioning the continuous value range to n partitions where the size of a partition is d. When tuning the parameter n for each vertex, we should make it an appropriately high value to minimize the loss of the continuity of data. However, making the n too high increase the overhead of learning and inference algorithms, therefore we should consider make it appropriately low value to make sure that algorithms can be run efficiently with the available resources.

$$g_{cd}(a_c) = \begin{Bmatrix} 0 \quad if \quad a_c < s_{a_c} \\ r \quad if \quad s_{a_c} + rd \leq a_c < s_{a_c} + (r+1)d \\ n-1 \quad if \quad a_c \geq e_{a_c} \end{Bmatrix} \tag{4.10}$$

The range for the continuous variables $((s_{a_c}, e_{a_c}))$ can be selected by checking the minimum and maximum values of the corresponding data values. However, that may consist of outliers that can make the range to be unnecessary wide. Therefore, it is recommended to use an outliers elimination algorithm to eliminate such outliers from the data and then take the minimum and maximum of the filtered data to decide the range. In our case studies, we used the standard deviation based outlier elimination technique [85] to evaluate the above range.

All the data values are mapped to discrete values using the above technique and a probability table are created for each node. The probability table of an independent vertex consists of n probability values, as it has to keep track of the probability of each of the discrete value of that variable. In contrast, a dependent vertex with m incoming edges keeps track of a probability table of $n^{(m+1)}$ entries corresponding to the all the possible combinations of the discreet values (values of m incoming vertices and the values of the current vertex).

Although there are advantages of using this technique such as the simplicity and ability to use of many discreet inferencing algorithms, it has many disadvantages such as the loss of continuity of data, and the low scalability. As the amount of memory to keep the probability tables and the processing power to carry out inferencing increases exponentially with the increase of variables (as message passing algorithms have to transform large probability tables [83]), this technique can not be used with complex Bayesian networks.

4.1.2.2 Regularized least squares regression

In the regularized least squares regression method, the context-QoS, context-context, QoS-QoS dependencies are trained as a linear regression problem. Here, the vertex (v_t) that have incoming edges from vertices ($V_s = v_1, v_2, .., v_n$) are written as a linear function of V_s. However, as the dependency relationship can be non-linear, we use basis functions (ϕ) to convert the elements in V_s to polynomial terms or multiplications of each other.

$$v_t = w^T \phi(V_s) \tag{4.11}$$

Here, the w is the set of parameters (corresponding to the weights of each elements of the basis function) to be trained by the regularized least squares regression algorithm.

To invoke the algorithm, the data points are divided in to training, validation and testing data sets conventionally with the ratios of 60% 20% and 20%. If V_s values of the training data set (after applying the basis function) is represented as a matrix Φ (where rows represent different data points and columns represent the terms of the basis function ϕ for each data point), and v_t values of the training data set is represented as a vector t, then the w can be evaluated using the following equation [58],

$$w = (\lambda I + \Phi^T \Phi)^{-1} \Phi^t t \tag{4.12}$$

Here, the λ is a regularization parameter (for a quadratic regularization term) to avoid over-fitting of the model to the training data. [58]. The parameter is tuned by minimizing the error with the set of validation data, and then the performance of the model is calculated using the testing data.

With this approach, the dependency functions of the Bayesian network can be represented as in the form of the equation 4.11. Compared with the tabular representation discussed in Section 4.1.2.1, this representation requires lesser memory. However, most of the Bayesian network inferencing algorithms can not be used with this approach. Only forward inferencing (i.e, infer the probabilities of child vertices, when all the parent vertices are known) can be performed efficiently. Additionally, this technique only provides the point values and not the probability distributions of the target variables. That is not useful in evaluating the trust of the some common QoS properties (such as response time, tracking error), which are distributions of values in nature.

4.1.2.3 Bayesian linear regression with sampling

Similar to regularized least square method in Section 4.1.2.2, the Bayesian linear regression also keeps the dependency function as a space and computational efficient equation. However, this equation used in this technique (equation 4.13) is a linear

Gaussian distribution resulting a probability distribution instead of a point value like in equation 4.11.

$$v_t = N\{m_N^T \phi(V_s), \sigma_N^2(V_s)\} \quad where \quad \sigma_N^2(V_s) = \frac{1}{\beta} + \phi(V_s)^T S_N \phi(V_s) \tag{4.13}$$

Similar to the regularized least square technique, the parameters m_N, and S_N would be trained using a set of training data using the equations 4.14, and 4.15. The parameters α, and β is tuned by trying out different values until the validation set gives a minimum error value.

$$m_N = \beta S_N \Phi^T t \tag{4.14}$$

$$S_N^{-1} = \alpha I + \beta \Phi^T \Phi \tag{4.15}$$

Although the direct use of the representation in equation 4.13 can be used only for forward inferencing with known independent variables, it is possible to use other sampling techniques such as forward sampling, rejection sampling and Gibbs sampling [58] to perform forward inferencing with some unknown independent variables and backward inferencing.

To perform forward inferencing with unknown independent variables using forward sampling, first, the independent vertices are sampled from a probability distributions chosen by domain experts to simulate the possible value ranges of the corresponding independent variables. For an example, the 'distance (d)' independent variable in the Bayesian network shown in Figure 4.3 can be sampled from a Uniform distribution with ranges of distance or from Gaussian distribution around a mean value for distance. Second, the child vertices are recursively sampled from Gaussian distributions with a deterministic mean and variance. (i.e., because the $\phi(V_s)$ in equation 4.13 can be deterministically evaluated from the parent's sample values). If some or all the independent variables are known, then we can use the known values as samples instead of sampling from probability distributions, and carry out the same process to generate samples for dependent variables. For an example, in the Bayesian network in Figure 4.3, if the distance(d), angle(a), resolution width(rw), and height(rh) is known, the forward sampling can be used to evaluate the tracked error(e) and response time(r). These evaluations are equivalent to the evaluation of $p(e|d, a, rw, rh)$ and $p(r|rw, rh)$.

If some of the dependent vertices are known, then we alter the forward sampling technique by rejecting samples that do not produce the values of the known dependent vertices. This altered technique is called rejection sampling. However, the main issue of using this method for continuous variable is, that it is very unlikely to generate a samples that have the exact known values. One possible solution is to use a range surrounding the known value as an acceptable value for the sample of the known dependent vertices. This could increase the probability of generating enough samples to do required inferences. The possible inferences that can be done with rejection sampling on the Bayesian network in Figure 4.3 are $p(d|e, a)$ and $p(e|d, a, rt)$.

If the network is more complex, it is possible that generating samples with known values for some vertices is very less probable for both discreet and continuous Bayesian networks. Gibbs sampling provides a technique to generate samples locally for each vertex (i.e., generate samples for each vertex depending only on its parents at a time). It can be used to generate samples for all the vertices of the Bayesian network iteratively. As the rejections have to be done only on local samples with Gibbs sampling, the probability of accepting samples are relatively higher. Therefore, Gibbs sampling technique can be used for inferencing complex Bayesian networks.

4.1.3 To Evaluate Trust from the Context-QoS Bayesian Network

After modeling the Bayesian network for a service (or a composed service), we will first find the cumulative distribution of the QoS property under the context that service is actually going to operate. 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 property by calculating the 'y'-axis value of cumulative distribution within the specification range. An example cumulative distribution for the response time (rt) of the camera service is shown in Figure 4.4. It also indicates the trust value corresponding to a selected specification of the response time.

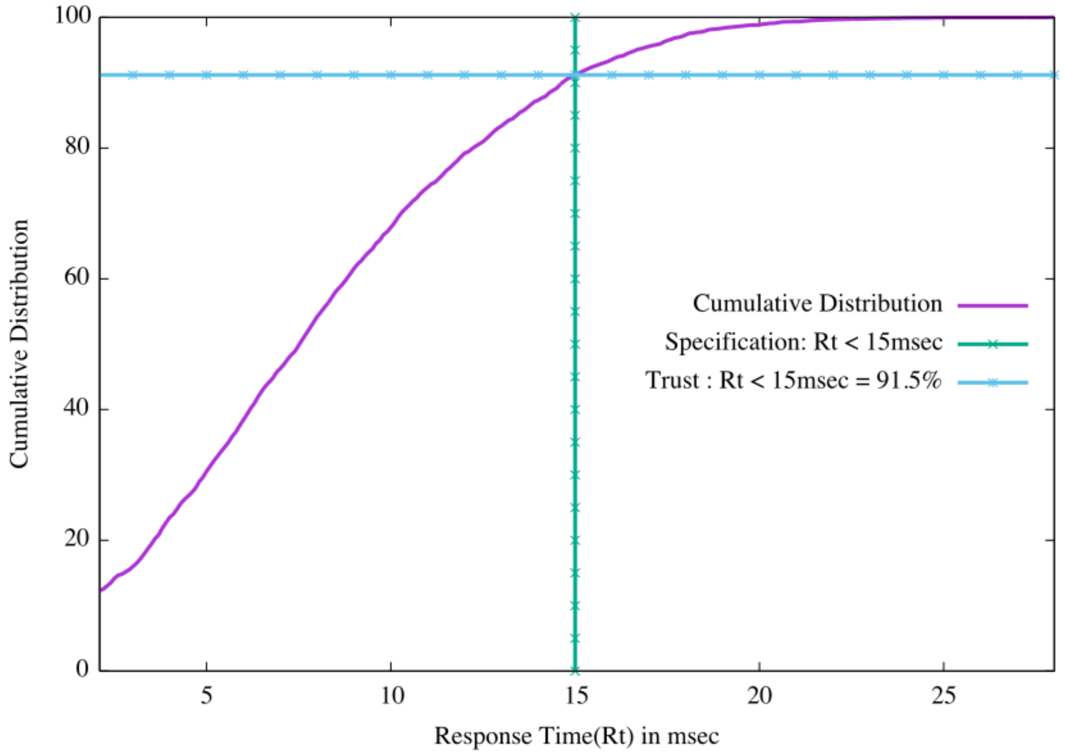


Figure 4.4.: Trust evaluation of the response time (rt) of camera tracking service of specification: $rt < 15msec$.

In the experiments shown in Chapter 5, 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.

4.1.4 Collecting Context-QoS Dependency Information for the Case Study

In this section, we build Bayesian networks of camera tracking service (Figure 4.1 and WiFi tracking service (Figure 4.2). For both of the services, we use the domain knowledge to create the structure of the Bayesian networks. The dependency functions of the networks are learned from the data from execution traces using the algorithms mentioned in the Section 4.1.2.

The structure of the Bayesian network designed for the camera tracking service is shown in Figure 4.3. Each data point used to train the Bayesian network consists of values of context properties (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 properties (i.e., the tracking error (e), and the response time (rt)). The equations in Table 4.2 indicate the dependency functions of the Bayesian network, when Bayesian linear regression with sampling technique is used. Note that the dependency function for e is an arithmetic operator instead of a linear Bayesian equation derived from a learning technique.

Table 4.2.: Dependency functions for the Bayesian network in Figure 4.3

Vertex	**Linear Gaussian Function**	
	Mean	**Variance**
rt	$0.52542w - 0.57251h + 0.0018238wh$	0.9
ex	$0.01545d + 52.988a - 0.0088268w + 0.018061h - 0.23193da - 0.00016626d^2 + 0.000022915d^2a^2$	0.7
ey	$-0.042133d + 12.654a - 0.0093142w + 0.018025h - 0.092207da - 0.000169d^2 + 0.000028905d^2a^2$	0.7
e	$\sqrt{(ex^2 + ey^2)}$	0

Similarly, the structure of the Bayesian network designed for the WiFi tracking service is shown in Figure 4.5. It can be shown that the response time of the service is independent of the context by representing it in a vertex without any incoming edges. The equations in Table 4.3 indicate the dependency functions of the Bayesian network, when Bayesian linear regression with sampling technique is used.

The results of the QoS and Trust inferences of the Bayesian networks of Camera tracking service and WiFi tracking service is presented in Section 5.1.1.

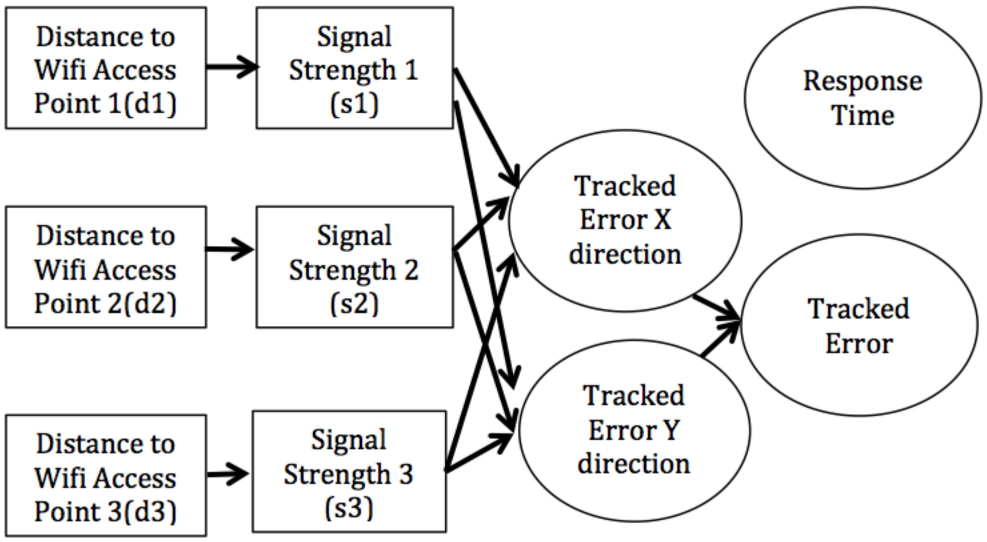

Distance to Wifi Access Point 1(d1)
Signal Strength 1 (s1)
Distance to Wifi Access Point 2(d2)
Signal Strength 2 (s2)
Distance to Wifi Access Point 3(d3)
Signal Strength 3 (s3)
Response Time
Tracked Error X direction
Tracked Error Y direction
Tracked Error

Figure 4.5.: The Bayesian network corresponding to the context-QoS dependencies of WiFi Tracking Service.

Table 4.3.: Dependency functions for the Bayesian network in Figure 4.5

Vertex	**Linear Gaussian Function**	
	Mean	**Variance**
$s1$	$-0.082514 - 0.17789d1 - 0.0039916d1^2$	21
$s2$	$-0.086065 - 0.19640d2 - 0.0018085d2^2$	19
$s3$	$-0.030455 - 0.18340d3 - 0.0013946d3^2$	20
ex	$-12.747s1 - 48.911s2 + 63.271s3 - 0.19723s1.s2 - 0.74909s2.s3 + 0.49537s3.s1 - 0.0098339s1.s2.s3$	7
ey	$-45.968s1 + 71.844s2 - 8.0355s3 - 0.36115s1.s2 + 0.99769s2.s3 + 0.19579s3.s1 + 0.0088851s1.s2.s3$	7
e	$\sqrt{(ex + ey)}$	0

4.2 Phase 2: To Collect Information about the Interaction Patterns in the 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 improve the QoS of the system. There are primitive interaction patterns [23], [17] as shown in list in Section 3.1.1. Complex interaction patterns are made by recursively combining these primitive patterns. We have shown in Section 3.1.4 that each interaction pattern has an associated operator that evaluate the QoS value of the composed system when we know the QoS values of the participating individual services.

4.2.1 Identifying Interaction Patterns and Composition Operators of the Case Study

In this sub-section, we use indoor tracking system composed out of services mentioned in table 4.1 as a case study. The design of the system is shown in Figure 4.6.

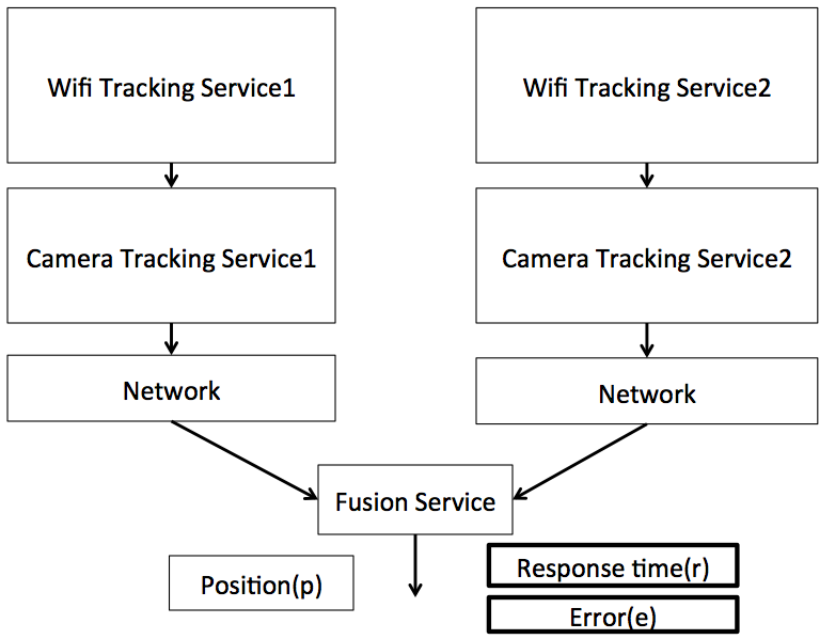


Figure 4.6.: Design of the indoor tracking system

The design in Figure 4.6 uses two interaction patterns from the list shown in Section 3.1.1.

1. **Sequence:** The camera tracking service (which track the position of the marker object relative to the camera), and the WiFi tracking service (which track the absolute position of the camera of the mobile phone) is interacting sequentially to get the absolute position of the object. Additionally, the network delay is

also represented as an another service that is interacting sequentially with the other services. The composed service out of these three services is denoted as s_a. Two instances of this service is created in the design in Figure 4.6 and they are denoted as s_{a_1}, and s_{a_2}.

2. **Split/ Join:** In order to get better prediction accuracy of position, the system averages out two independently predicted positions from s_{a_1} and s_{a_2} using a fusion service. The resultant composed service is equivalent to the complete tracking system which is denoted as s_b.

We are interested in two QoS properties of the composed system, response time and tracking error. Composition operators for the response time can be looked up from the Tables 3.1 and 3.2. Therefore, addition operator is used for the sequence interaction pattern and the maximum operator is used for the parallel split/join pattern.

Since the tracking error is not a common property listed in the Tables 3.1 and 3.2, we have to derive the necessary composition operators from the domain knowledge of the tracking system. When calculating the absolute position of an object marker for s_a, we add the position coordinates (x_c^p, y_c^p) predicted by camera tracking service (s_c) to the position coordinates (x_w^p, y_w^p) predicted by WiFi tracking service(s_w). If the actual position of the marker relative to the camera is (x_c^a, y_c^a) and the mobile phone with the camara is (x_w^a, x_w^a), then the difference between the predicted absolute position and the actual absolute position of the object marker is $(diff_{p-a})$,

$$\begin{aligned} diff_{p-a} &= (x_c^p + x_w^p, y_c^p + y_w^p) - (x_c^a + x_w^a, y_c^a + y_w^a) && (4.16) \\ &= (x_c^p - x_c^a + x_w^p - x_w^a, y_c^p - y_c^a + y_w^p - y_w^a) && (4.17) \\ &= (x_c^p - x_c^a, y_c^p - y_c^a) + (x_w^p - x_w^a, y_w^p - y_w^a) && (4.18) \\ \tilde{e_a} &= \tilde{e_c} + \tilde{e_w} && (4.19) \end{aligned}$$

Here, the terms $\tilde{e_a}$, $\tilde{e_c}$, and $\tilde{e_w}$ terms are error vectors of the services s_a, s_c, and s_w. Since we keep track of the errors in both 'x' direction and 'y' direction, we will be able to calculate the error of the composed system using the equation 4.19.

Similarly, it can be shown that we can use the following operator as the composition operator of the tracking error for the parallel split/join interaction pattern.

$$e_b = \frac{\tilde{e_{a_1}} + \tilde{e_{a_2}}}{2} \tag{4.20}$$

The summary of the operators used in the case study is indicated in Table 4.4.

Table 4.4.: Operators used in the case study for indoor tracking system

QoS	**Operators for Sequence pattern**	**Operators for Parallel Join Pattern**
Response Time	Addition	Maximum
Error (X & Y)	Addition	Mean

4.3 Phase 3: To Derive Context-QoS Bayesian Networks for Compositions

After building the Bayesian networks for individual services and identifying the composition operators, we derive the Bayesian network for the composed system (referred as 'Composed Bayesian network') using the following rules:

- Bayesian networks corresponding to the individual services will be part of the composed Bayesian network as sub-networks.
- If two services share a context, then the composed Bayesian network will have one vertex representing the shared context.
- There will be a new QoS vertex for each interaction between services (corresponding to the composed QoS property), new edges connecting the new QoS

vertices to other QoS properties in the networks (that the new QoS depends on), and dependency functions corresponding to the composition operators.

Formally, if there are services $S_1, S_2, .., S_n$; each service $S_r \quad (r = 1, 2, ..n)$ has an associated Bayesian network $G_{s_r} = (V_{s_r}, E_{s_r}, F_{s_r})$; services interactions are denoted as $(I = i_1, i_2, ..i_m)$; then the Bayesian network for the composed system S_c (which is denote as $G_c = (V_c, E_c, F_c)$) can be derived as following:

$$V_c = V_{s_1} \cup V_{s_2} \cup ...V_{s_n} \quad \cup \quad Q_{i_1} \cup Q_{i_2} \cup ... \cup Q_{i_m} \tag{4.21}$$

$$E_c = E_{s_1} \cup E_{s_2} \cup ...E_{s_n} \quad \cup \quad E_{i_1} \cup E_{i_2} \cup ... \cup E_{i_m} \tag{4.22}$$

$$F_c = F_{s_1} \cup F_{s_2} \cup ...F_{s_n} \quad \cup \quad F_{i_1} \cup F_{i_2} \cup ... \cup F_{i_m} \tag{4.23}$$

Here, the Q_{i_x} represents the set of composed QoS properties with the interaction of services i_x, the E_{i_x} represents the set of edges coming into Q_{i_x} from the vertices form the $(V_{s_1} \cup V_{s_2} \cup ...V_{s_n})$, and the F_{i_2} represents the composition operator at interaction i_x.

With the proposed approach, both the structure and the parameters of the composed Bayesian network can be derived from the information from the Bayesian networks of participating services and the composition operators. Therefore, it is not required to have data to train the parameters and the structures of the composed Bayesian network. This is a major advantage of this model, as with the derived composed Bayesian network, we can simulate the QoS and Trust behaviour of the network, before even we build the composed system.

4.3.1 Deriving Composed Bayesian Network for the Case Study

The Bayesian network for composition of a WiFi tracking service and a Camera tracking service with sequence interaction pattern is shown in the Figure 4.7.

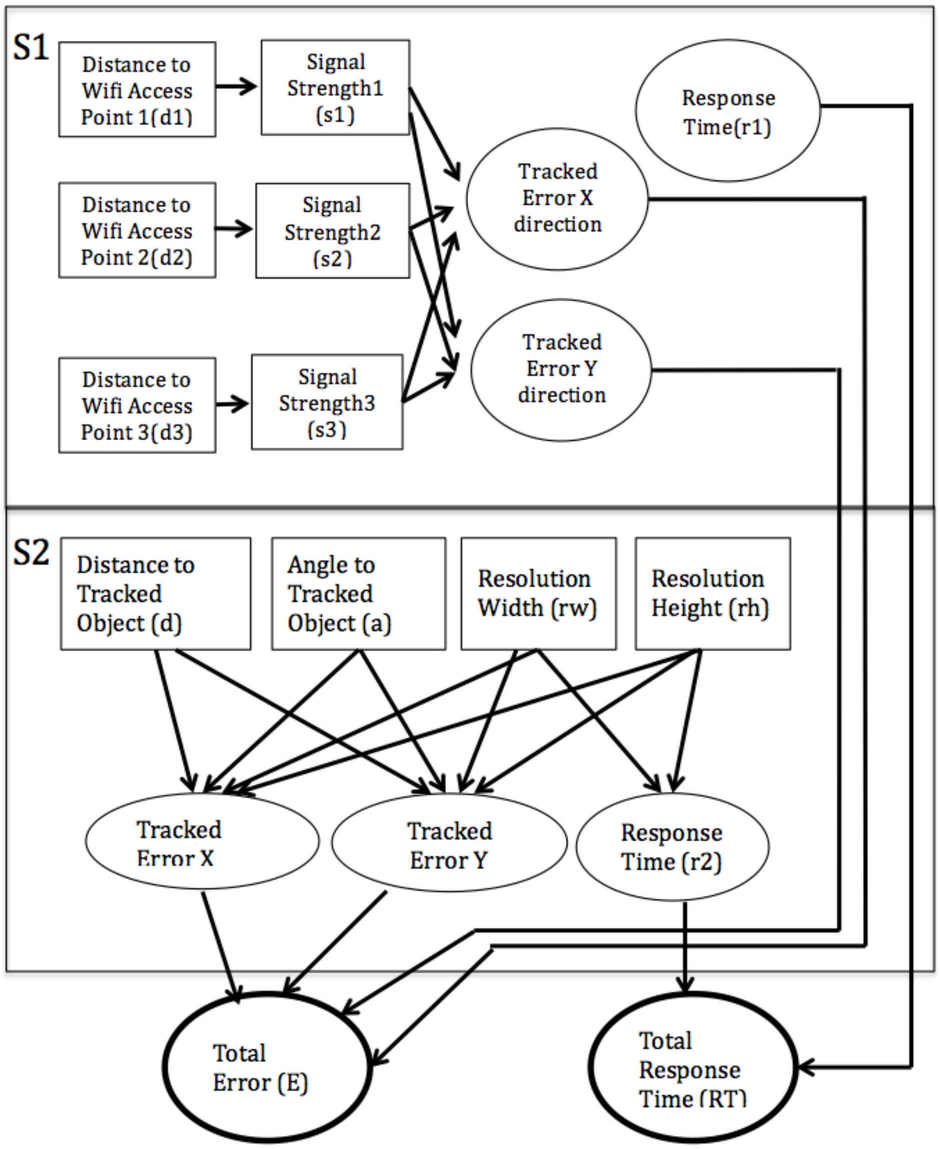


Figure 4.7.: Bayesian network of the composed service of WiFi tracking service and camera tracking service.

Here, there is one interaction pattern, namely the sequence interaction pattern. Total response time (tr), and the total error (te), which are the QoS properties associated with the interaction patterns, have become the vertices of the composed Bayesian network. New edges are added from QoS vertices of the Bayesian networks corresponding to individual services to the composed QoS properties. The composition operators are used as dependency functions (in addition to the dependency functions coming from Bayesian networks of individual services) of the composed Bayesian network.

4.4 Phase 4: Use Inference Techniques to Answer QoS/Trust Queries

After creating the composed Bayesian network, it will be used to carry out inferencing about QoS and trust behaviours of the composed system. The inference techniques used for the composed Bayesian network are similar to the inference techniques described for Bayesian network of individual services in Sub-section 4.1.2. Following types of inferencing can be performed on the composed Bayesian network:

- To predict QoS of the composition using the available context information – (forward inferences).
- To predict the trust of the composed system using the available context information – (forward inferences).
- To evaluate the context properties that will provide required QoS for the composed system – (Backward inferences).
- To evaluate QoS and trust of reused participating services using the existing composed systems – (Backward inferences).

We describe the examples of forward inferencing to predict QoS and trust of composed system with case studies in Chapter 5 and applications of the model in selecting optimum subset of services in Section 6.1.1. Similarly, the use of backward inferencing is described in Section 6.2 and in Section 6.3.

In this chapter, we have presented the ‘ContextTrust’ model, which acts as the basic prediction model of the context dependent QoS and trust prediction framework. This model is capable of predicting QoS and trust of composed systems using the information available at design phase of the system development lifecycle. We divide the evaluation process into four 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 fourth, perform inferencing about trust and QoS of the composed system using the final Bayesian network. We presents the effectiveness of the proposed model, by carrying out an empirical validation using case studies, in Chapter 5.

5 CASE STUDIES

This chapter provides case studies to validate the ContextTrust model described in Chapter 4. The content in this chapter is an extension of our previous publications [19, 20]. We use case studies from three different domains as listed below.

1. Indoor tracking system
2. Travel planning system
3. Collaborative bullying classification system

Here, the Indoor tracking system uses services developed by our research group [79] for internal experimentations. The travel planning system mainly uses publicly available services from different vendors. Collaborative bullying classification system uses standard machine learning techniques like Naive Bayes, Logistic Regression, and Support Vector machines as participating services. Therefore, by selecting the above case studies, we expect to validate our model with services developed with different development practices.

To validate effectiveness of the ContextTrust model, we evaluate the errors associated with the QoS and trust inferences of these systems using the model and compare the results with the errors related to the QoS and trust evaluated using the prevalent technique proposed by Hwang et al. [23] that do not consider context-QoS dependencies. The error values associated with the results of the inferences are calculated using the relative absolute error metric [86], which is defined in equation 5.1. Each case study and the corresponding results are described in detail in different sections of this chapter.

$$relative \quad absolute \quad error = \frac{\sum |predicted\_value \quad - \quad actual\_value|}{\sum |actual\_value\_mean \quad - \quad actual\_value|} \tag{5.1}$$

5.1 Indoor Tracking System

The Indoor tracking system is used to explain the application of ContexTrust model in Chapter 4. The participating services of the system are listed in Table 4.1. During the discussion about the model, we develop Bayesian networks for individual services (Camera Tracking Service and WiFi Tracking Service) and composed system. Here we present the results of QoS and trust inferences on Bayesian networks of individual services (Sub-section 5.1.1), and composed system (Sub-section 5.1.2).

5.1.1 Results of Inferencing on a Bayesian Network of a Single Service

The algorithms discussed in section 4.1.2 are applied to the Bayesian network corresponding to the camera tracking service (represented in Figure 4.3). The resultant errors of each of the three algorithms are shown in Table 5.1. The table also shows the error for the predictions performed using mean values of the training data, which assume that the QoS 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. The 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 machine learning techniques provide significantly accurate predictions. Since the response time is not significantly dependent on context, its predictions are closer to the mean response time. From here on, we use Bayesian linear regression with sampling

Table 5.1.: Relative absolute errors of forward inferencing of single service Bayesian network

Algorithm	**Relative Absolute Error of** $(e\|d, a, rw, rh)$	**Relative Absolute Error of** $(rt\|rw, rh)$
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

to perform additional inferencing such as trust evaluations on individual services and composed systems.

After evaluating the QoS, the trust of the service can be evaluated by analyzing the resultant QoS distribution. However, if the service is used in a context that the context attributes can take any value 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. With the restricted context, the trust value as a percentage for different tracking error specification values is listed in the Table 5.2.

Table 5.2.: Trust evaluation with and without consideration of context for the camera tracking service

Specification of tracking error	3cm	4cm	5cm	6cm	7cm
Predicted Trust (without considering context)	43.901	63.334	79.724	90.790	96.518
Predicted Trust (with considering context)	62.909	86.132	95.744	97.866	99.941
Percentage that actually met the specification	67.293	91.862	99.531	99.844	100

The graph in Figure 5.1 shows the curves for predicted trust values without considering the context (prevalent approach), predicted trust value with considering the context (our approach), and the actual percentage that met the specification against different choice of specification values. From the Figure 5.1 and Table 5.2, 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 in a restricted context, where as our approach that considers the context gives comparatively accurate predictions.

Similar inferences are performed with the Bayesian network corresponding to the WiFi tracking service shown in Figure 4.5. The relative absolute errors of different approaches are show in the Table 5.3.

To evaluate trust predictions, we used a restricted context ($d1 \geq 100, d2 \leq 300$) for the WiFi tracking service. In this restricted context, the trust predictions for the error of the WiFi tracking service (for selected specification values for error) using different approaches is shown in the Table 5.4. The predictions are shown visually in the following graph (Figure 5.2) for a wide range of error specification values.

These results confirm that the QoS and trust prediction techniques that consider the context provides more accurate results than the prediction techniques that does not consider the context.

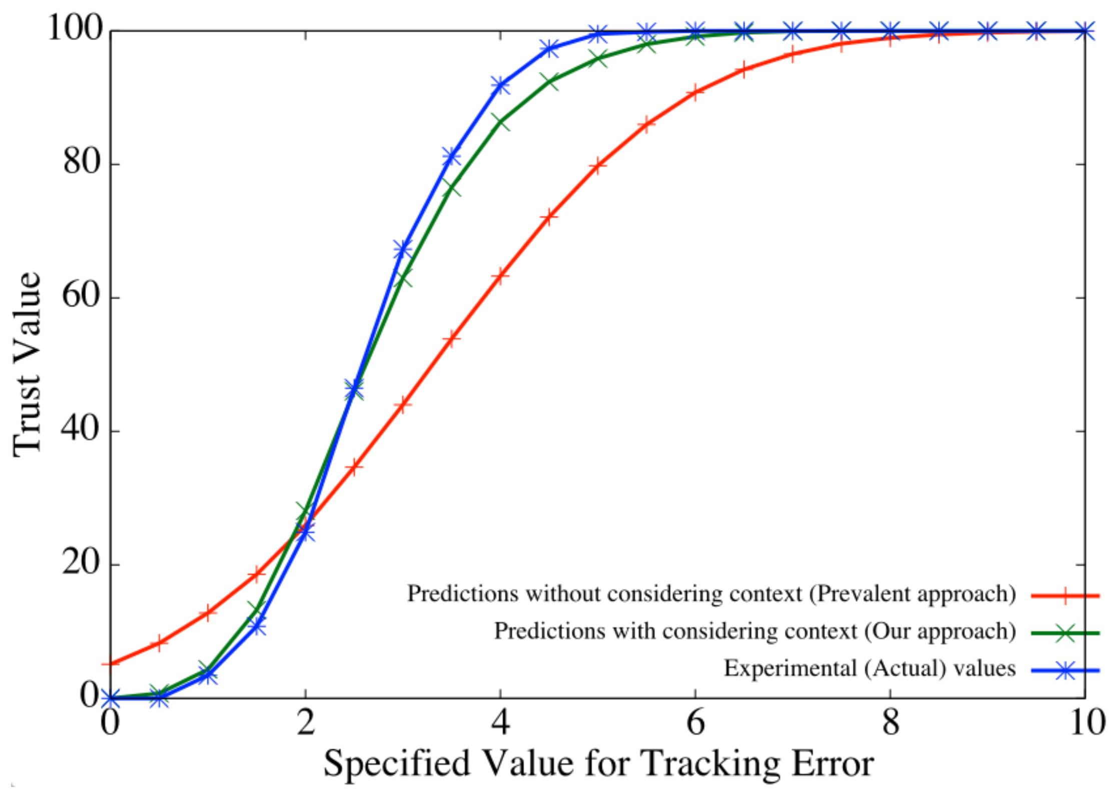


Figure 5.1.: A graph showing the trust of the error using different approaches for different choice of specification values (for camera tracking service).

Table 5.3.: Relative absolute errors of forward inferencing of single service Bayesian network

Algorithm	**Relative Absolute Error of** $(e\|d1, d2, d3)$
Message passing algorithm	0.463
Regularized least squares regression	0.292
Bayesian Linear regression with sampling	0.310
From the means of training data	1.08

5.1.2 Results of Inferencing on a Bayesian Network of a Composition

Table 5.5 shows the relative absolute error of the predictions performed using the Bayesian network approach (which consider the context) and the prevalent approach [23] which do not consider context in their predictions.

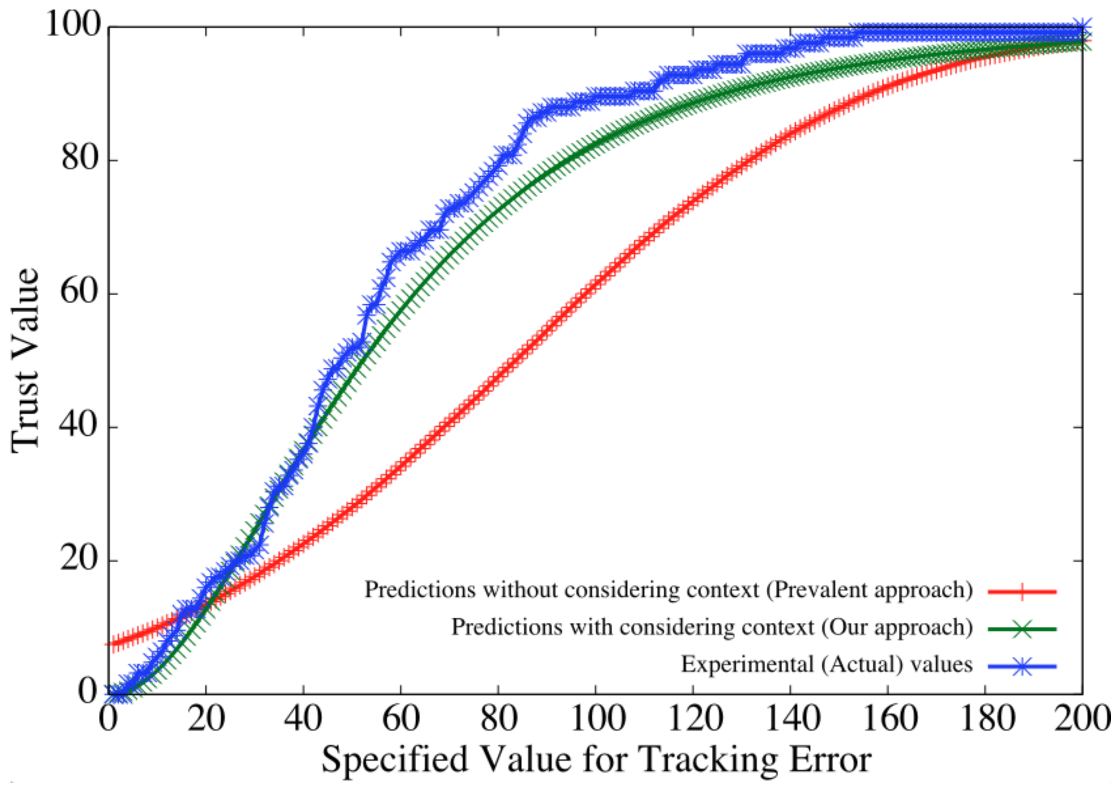


Figure 5.2.: A graph showing the trust of the error using different approaches for different choice of specification values (for WiFi tracking service).

Table 5.4.: Trust evaluation with and without consideration of context for the WiFi tracking service

Specification of tracking error (cm)	80	100	120	140	160
Predicted Trust (without considering context)	47.438	61.25	73.7	83.416	91.017
Predicted Trust (with considering context)	72.53	82.33	88.77	92.224	94.889
Percentage that actually met the specification	79.2	89.66	93.6	96.8	99.2

Table 5.5 shows that considering the context gives more accurate predictions for the tracking error compared with the prevalent approach. However, the quality of the predictions for the response time has improved only slightly by considering the

Table 5.5.: QoS predictions of service composition

Approach	**Relative absolute error of predictions of response time**	**Relative absolute error of predictions of tracking error**
Bayesian network based approach (Considering the context)	0.939	0.408
Prevalent approach (Without considering the context)	1.249	1.142

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.

Similarly, the trust predictions can be obtained by forward sampling from the composed Bayesian network. Similar to the experiments performed on a single service (Table 5.2), we use the following restrictions for the context.

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 the restricted context, the trust values for different tracking error specification values is listed in the Table 5.6 and visualized in the graph Figure 5.3. The graph shows the proposed approach provides highly accurate QoS & trust predictions of composed services specially if we know the restrictions of the context of the system.

5.2 Travel Planning System

The service groups and the corresponding candidate services that the travel planning system is composed of are listed in the Table 3.6. The candidate services include

Table 5.6.: Trust evaluation with and without consideration of context for composed tracking system

Specification of the tracking error (cm)	60	80	100	120	140
Predicted Trust (without considering context)	38.413	60.446	78.374	90.099	96.034
Predicted Trust (with considering context)	64.085	84.422	94.055	98.038	99.453
Percentage that actually met the specification	62	81.333	90	97.333	98.667

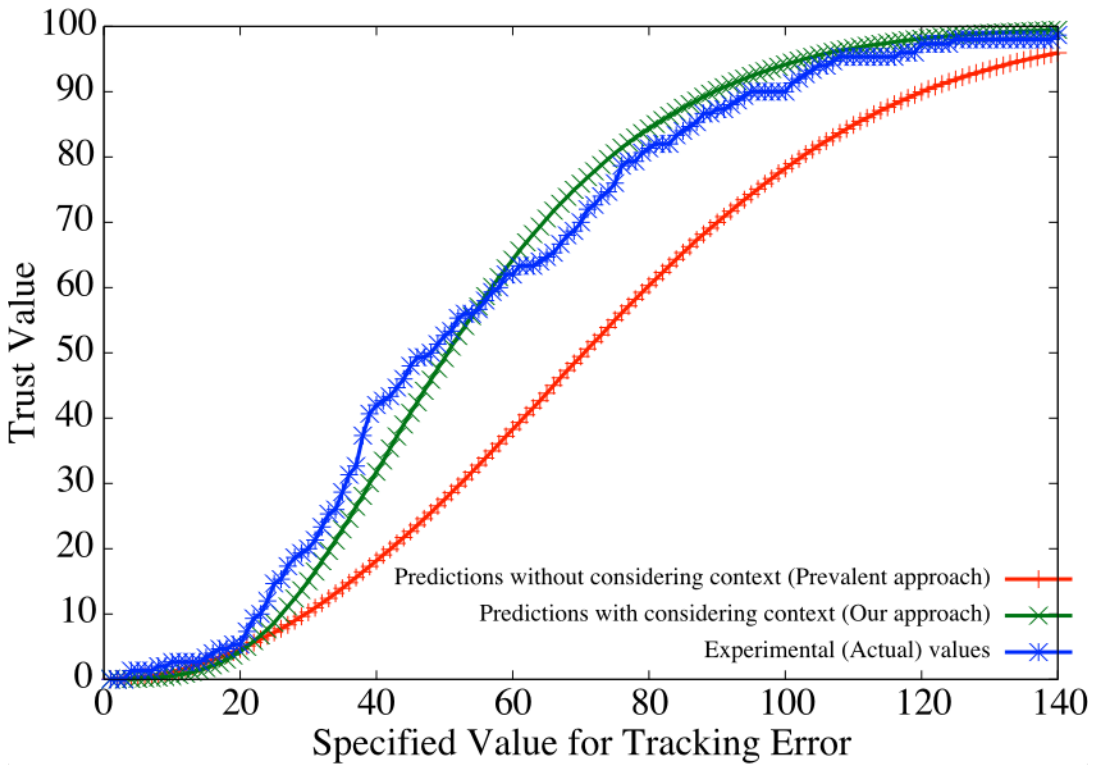


Figure 5.3.: A graph showing the trust of the error using different approaches for different choice of specification values (for composed tracking system).

public services as well as custom written services. Some of the custom services are written to share the same SOAP session [87] with the services in other groups to emulate associations between them. We have used the same set of services in our previous work (described in section 3.2.2) to identify the association between publicly available services.

In this case study we have altered the services to improve its security properties. 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 enforce security, 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 massages enforced (Sign).
4. An endpoint with encrypting of the messages enforced (Enc).

Wrapper services are implemented and deployed using Apache Axis2 Web service engine [88] with Apache Rampart [89] and Apache Neethi [90] 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 approximately in a linear function (with more directions and traffic information), and the time to encrypt and sign the messages also increases. Similarly, the response times of the Hotel, Weather and Car Rental services depend on all the security configurations

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 studies described in Section 3.2.2, 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 5.4.

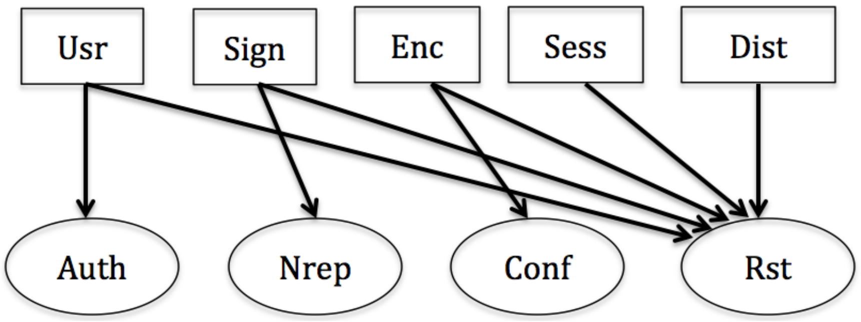


Figure 5.4.: Bayesian network of the direction/ traffic services

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 5.5. After deducing the structure of the Bayesian network, the dependency 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 properties using the Table 3.1. The operators are separately listed in the Table 5.7.

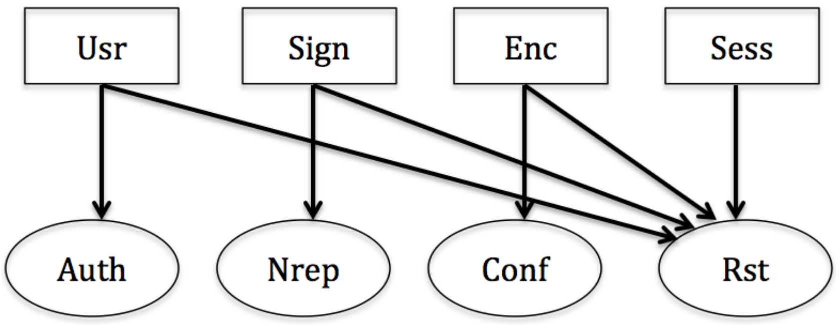


Figure 5.5.: Bayesian network of the hotel/ weather/ car rental services

Table 5.7.: Operators used in the travel planning system case study

QoS	**Operators for Sequence pattern**
Response Time (Rst)	Addition
Authentication/ Authorization (Auth)	Universal
Authentication/ Non-repudiation Nrep	Universal
Confidentiality (Conf)	Universal

The Bayesian network corresponding to a candidate composition is derived by aggregating the Bayesian networks of the participating services as described in Section 4.3. Figure 5.6 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 a addition, we would arithmetically calculate the composed response time (by adding means and variance separately) [61], instead of forward sampling on the composed Bayesian network.

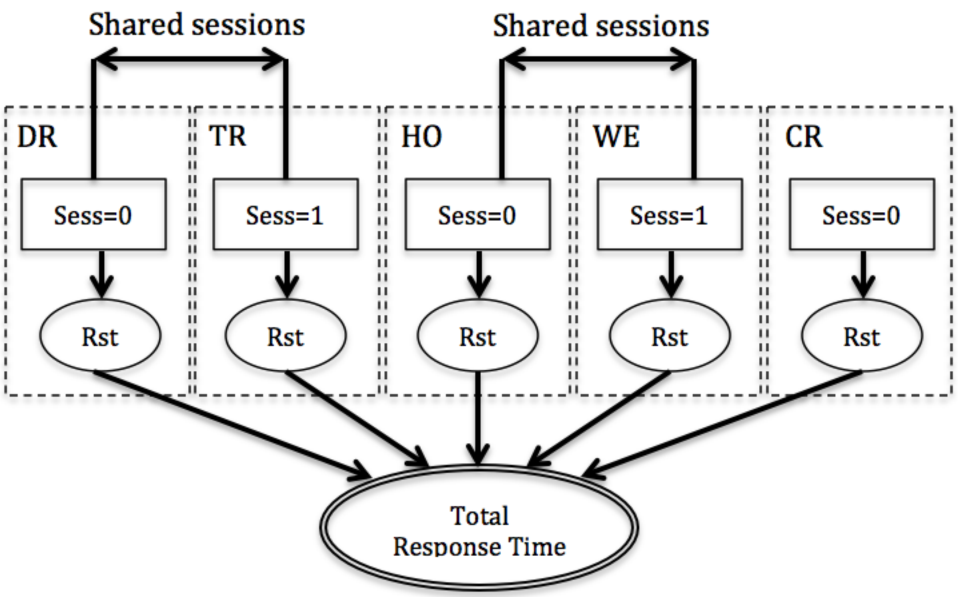


Figure 5.6.: Bayesian network of a candidate composition

From the set of possible compositions, we use the composition with services DR1, TR1, HO3, WE2, CR1 to graphically shows the results of the 'ContextTrust' model.

When the context is $Dist = 200miles, Usr = true, Sign = false, Enc = false$ (referred as 'Travel-Context-A'), the trust of the composition predicted by both the prevalent approach and our approach and the actual values are listed in Table 5.8 and graph in Figure 5.7.

When the context is $Dist = 3000miles, Usr = false, Sign = false, Enc = true$ (referred as 'Travel-Context-B'), the trust of the composition predicted by both the prevalent approach and our approach and the actual values are listed in Table 5.9 and graph in Figure 5.8.

The reason we choose to show the results of the two contexts 'Travel-Context-A' and 'Travel-Context-B' is 'Travel-Context-A' causes a significantly lower response time of the system compared to the response time caused from 'Travel-Context-B'. The graphs in Figure 5.8 and Figure 5.7 highlight that the proposed ContextTrust closely predicts the distributions of the response time for two different contexts separately, where as context independent trust model use the same predictive distribution

Table 5.8.: Trust evaluation with and without consideration of context for travel planning system with 'Travel-Context-A'

Specification of Response Time (ms)	1200	1600	2000	2400	2800
Predicted Trust (without considering context)	2.710	4.805	7.951	12.730	19.064
Predicted Trust (with considering context)	20.914	45.352	71.737	89.767	97.498
Percentage that actually met the specification	6.5	39.5	83	98.5	100

Table 5.9.: Trust evaluation with and without consideration of context for travel planning system with 'Travel-Context-B'

Specification of Response Time(ms)	6000	7000	8000	9000	10000
Predicted Trust (without considering context)	88.897	96.907	99.370	99.923	99.99
Predicted Trust (with considering context)	0.002	1.209	29.892	88.502	99.827
Percentage that actually met the specification	0	4.5	31	75	95.5

and it is failed to predict the two special cases of the contexts. Therefore, we can conclude that the Context-Dependent model is more effective in predicting QoS and Trust of applications that have high variance of QoS depending on the context.

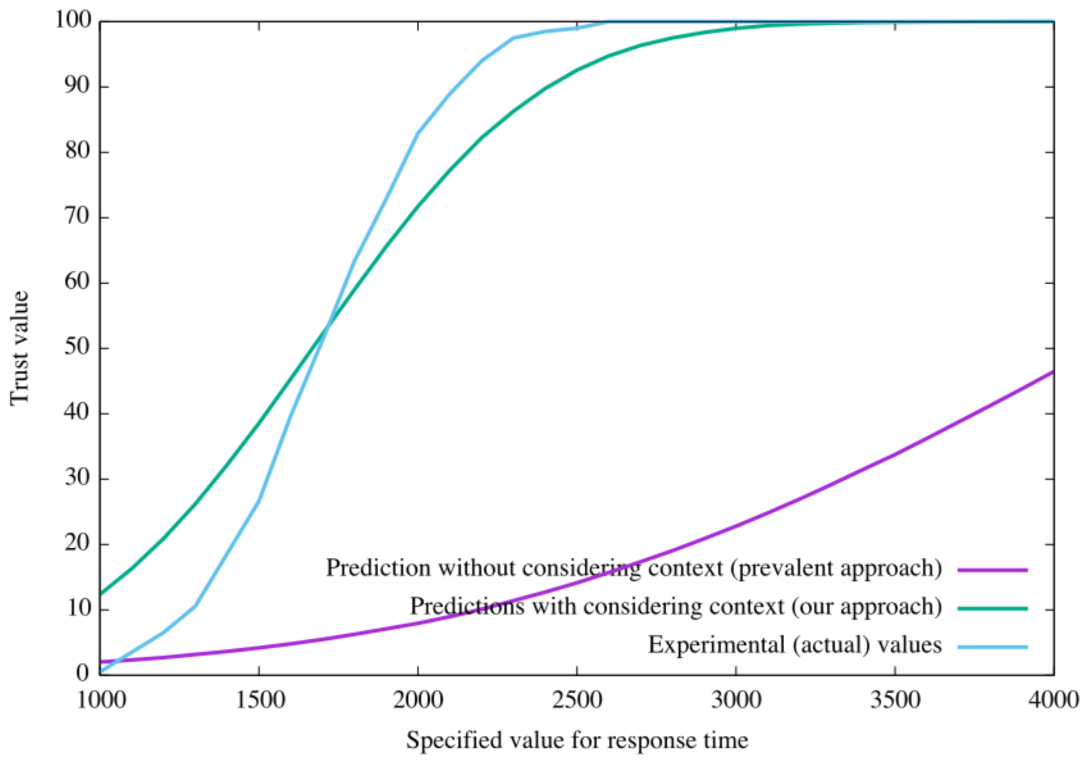


Figure 5.7.: A graph showing the trust of the response time using different approaches for different choice of specification values (for travel planning system with 'Travel-Context-A').

5.3 Collaborative Bullying Classification System

Collaborative bullying classification system [91] classifies tweets in a twitter feed either as bullying or non-bullying. The system uses three bullying classification services based on the following machine learning algorithms.

1. Naive Bayes classifier (NB)

2. Logistic Regression classifier (LR)

3. Support Vector Machine classifier (SVM)

Each of these services have QoS properties of precision (PR), and recall (RC) [92]. They represent the aggregated performance of the classifiers associated with a set of testing data. These properties are calculated using True Positives(TP), False Posi-

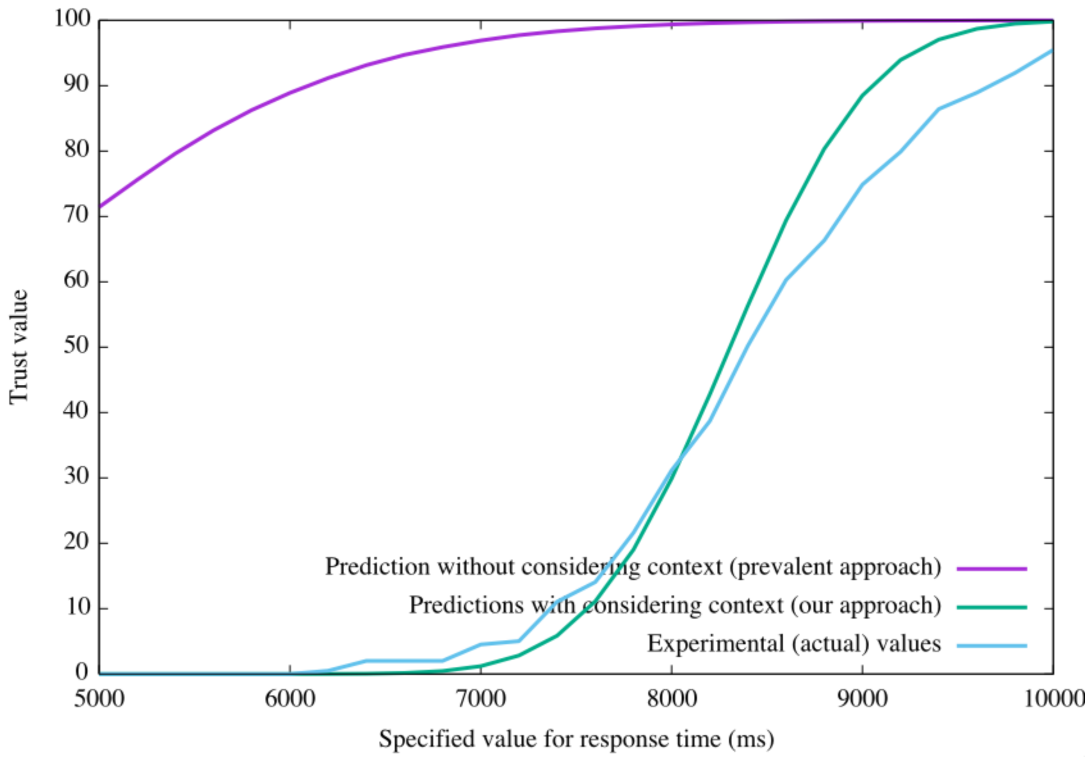


Figure 5.8.: A graph showing the trust of the response time using different approaches for different choice of specification values (for travel planning system with 'Travel-Context-B').

tives(FP), True Negatives(TN), and False Negatives (TN) as shown in equations 5.2, and 5.3.

$$PR = \frac{TP}{TP + FP} \tag{5.2}$$

$$RC = \frac{TP}{TP + FN} \tag{5.3}$$

To represent these values as probability distributions, we use the Beta distribution as shown in equations 5.5. As the Beta distribution requires non-zero parameters, when any of TP, FP, or FN is equal to zero, we approximate it with the value 0.1.

$$PR \approx \beta(TP, FP) \tag{5.4}$$

$$RC \approx \beta(TP, FN) \tag{5.5}$$

The precision and recall depend on the biasness of the data. We identified three different domains that have three different biasness with respect to data as being bullying or not. These domain are politics, sports, and education. With the above mentioned domain knowledge, the Bayesian networks of each classifier services can be built as in Figure 5.9.

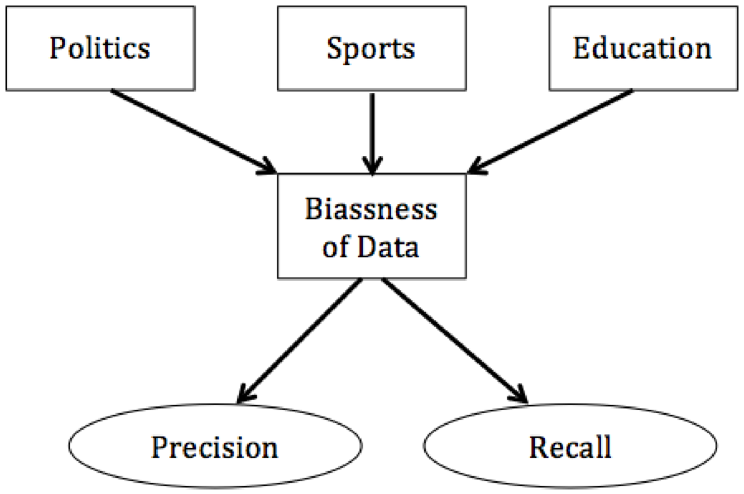


Figure 5.9.: Bayesian network for the classifier services

From each of the domain, we have collected tweets by two methods. First searching a keyword related to the domain, and second retrieving tweets from a personal twitter account related to the domain. The keywords and the account for each domain one shown in Table 5.10 along with the short-notation used for later references. Additionally, aggregated tweets related to politics, sports, and education are referred as 'PL*', 'SP*', and 'ED*'. Aggregation of all the tweets are referred are referred as 'ALL'.

Table 5.10.: Twitter data sources for each domain

Domain	**Keywords**	**Personal Account**
Politics	Election (PL1) Obama (PL2) Clinton (PL3) Trump (PL4)	@BarackObama (PL5) @realDonaldTrump (PL6) @HillaryClinton (PL7)
Sports	Olympic (SP1) Bolt (SP2) Phelps (SP3)	@MichaelPhelps (SP4) @usainbolt (SP5)
Education	Education (ED1) coolcatteacher (ED2) kevin_corbettab (ED3)	@coolcatteacher (ED4) @kevin_corbettab (ED5)

The precision and recall values for each twitter feed (and the aggregated feeds for each domain) is shown it Table 5.11 as mean values and Table 5.12 as beta distributions. Here the 'NaN' represents 'Not a Number'.

The classifier services are composed with two different collaboration patterns as following (The generic pattern of the composed system is shown in Figure 5.10).

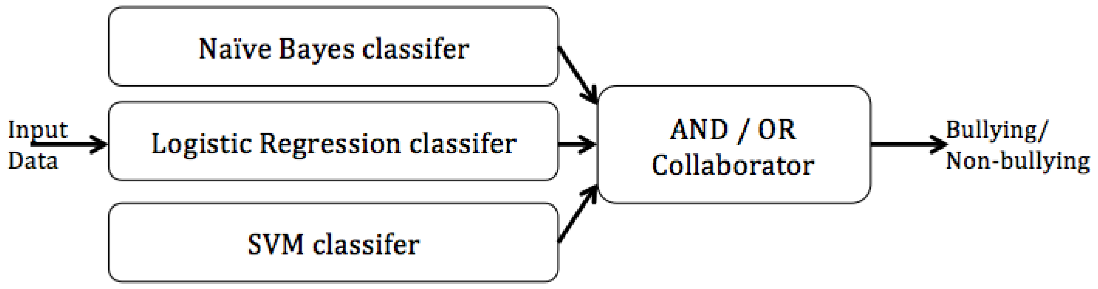


Figure 5.10.: Generic collaboration pattern of the bullying classification system

1. 'OR' Collaboration: The system classifies a data instance as bullying, only if one of the classifier services classifies the data instance as bullying.

Table 5.11.: QoS (PR,RC) of the classifier services for different twitter feeds

Feed	**SVM (PR, RC)**	**LR (PR, RC)**	**NB (PR, RC)**
PL1	(73, 46)	(70, 75)	(43, 66)
PL2	(99, 81)	(98, 88)	(97, 90)
PL3	(96, 78)	(96, 88)	(92, 88)
PL4	(89, 61)	(87, 81)	(83, 82)
PL5	(100, 75)	(58, 69)	(50, 81)
PL6	(96, 59)	(83, 72)	(71, 80)
PL7	(92, 48)	(82, 63)	(75, 85)
PL*	(94, 66)	(87, 79)	(78, 84)
SP1	(100, 14)	(15, 57)	(28, 28)
SP2	(NaN, 0)	(12, 20)	(0, 0)
SP3	(NaN, 0)	(11, 50)	(0, 0)
SP4	(NaN, 0)	(0, 0)	(6, 50)
SP5	(NaN, 0)	(0, 0)	(6, 100)
SP*	(100, 5)	(8, 35)	(8, 25)
ED1	(50, 7)	(22, 38)	(13, 15)
ED2	(NaN, 0)	(20, 100)	(0, 0)
ED3	(NaN, NaN)	(NaN, NaN)	(NaN, NaN)
ED4	(NaN, 0)	(0, 0)	(0, 0)
ED5	(NaN, 0)	(7, 16)	(7, 16)
ED*	(50, 4)	(13, 31)	(7, 13)
ALL	(93, 63)	(74, 77)	(70, 80)

2. 'AND' Collaboration: the system classifies the data instance as bullying, only if all the classifier services classify the data instance as bullying.

There do not exist simple exact operators for the QoS aggregation of the composed system in both patterns. But there are approximate composition operators that we can use to calculate the TP, FP, and FN for both collaboration patterns. Evaluations of these values of the composed system allow us to evaluate its precision and recall as Beta distributions. The operators corresponding to each QoS and the collaboration pattern are shown in Table 5.13.

We apply these operators to predict the QoS of collaboration pattern. Since the context values are discrete, we will use a tabular method to infer the QoS and Trust

Table 5.12.: QoS distributions of the classifier services for different twitter feeds

Feed	**SVM (PR, RC)**	**LR (PR, RC)**	**NB (PR, RC)**
PL1	$\beta(28,10)$, $\beta(28,32)$	$\beta(45,19)$, $\beta(45,15)$	$\beta(40,51)$, $\beta(40,20)$
PL2	$\beta(131,1)$, $\beta(131,30)$	$\beta(143,2)$, $\beta(143,18)$	$\beta(145,3)$, $\beta(145,16)$
PL3	$\beta(118,4)$, $\beta(118,32)$	$\beta(132,5)$, $\beta(132,18)$	$\beta(133,11)$, $\beta(133,17)$
PL4	$\beta(83,10)$, $\beta(83,51)$	$\beta(109,15)$, $\beta(109,25)$	$\beta(110,22)$, $\beta(110,24)$
PL5	$\beta(25,0.1)$, $\beta(25,8)$	$\beta(23,16)$, $\beta(23,10)$	$\beta(27,27)$, $\beta(27,6)$
PL6	$\beta(56,2)$, $\beta(56,38)$	$\beta(68,13)$, $\beta(68,26)$	$\beta(76,31)$, $\beta(76,18)$
PL7	$\beta(46,4)$, $\beta(46,49)$	$\beta(60,13)$, $\beta(60,35)$	$\beta(81,27)$, $\beta(81,14)$
PL*	$\beta(487,31)$, $\beta(487,240)$	$\beta(580,83)$, $\beta(580,147)$	$\beta(612,172)$, $\beta(612,115)$
SP1	$\beta(1,0.1)$, $\beta(1,6)$	$\beta(4,22)$, $\beta(4,3)$	$\beta(2,5)$, $\beta(2,5)$
SP2	$\beta(0.1,0.1)$, $\beta(0.1,5)$	$\beta(1,7)$, $\beta(1,4)$	$\beta(0.1,3)$, $\beta(0.1,5)$
SP3	$\beta(0.1,0.1)$, $\beta(0.1,4)$	$\beta(2,16)$, $\beta(2,2)$	$\beta(0.1,3)$, $\beta(0.1,4)$
SP4	$\beta(0.1,0.1)$, $\beta(0.1,2)$	$\beta(0.1,12)$, $\beta(0.1,2)$	$\beta(1,15)$, $\beta(1,1)$
SP5	$\beta(0.1,0.1)$, $\beta(0.1,2)$	$\beta(0.1,15)$, $\beta(0.1,2)$	$\beta(2,27)$, $\beta(2,0.1)$
SP*	$\beta(1,0.1)$, $\beta(1,19)$	$\beta(7,72)$, $\beta(7,13)$	$\beta(5,53)$, $\beta(5,15)$
ED1	$\beta(1,1)$, $\beta(1,12)$	$\beta(5,17)$, $\beta(5,8)$	$\beta(2,13)$, $\beta(2,11)$
ED2	$\beta(0.1,0.1)$, $\beta(0.1,1)$	$\beta(1,4)$, $\beta(1,0.1)$	$\beta(0.1,3)$, $\beta(0.1,1)$
ED3	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$
ED4	$\beta(0.1,0.1)$, $\beta(0.1,2)$	$\beta(0.1,11)$, $\beta(0.1,2)$	$\beta(0.1,8)$, $\beta(0.1,2)$
ED5	$\beta(0.1,0.1)$, $\beta(0.1,6)$	$\beta(1,12)$, $\beta(1,5)$	$\beta(1,13)$, $\beta(1,5)$
ED*	$\beta(1,1)$, $\beta(1,21)$	$\beta(7,44)$, $\beta(7,15)$	$\beta(3,37)$, $\beta(3,19)$
ALL	$\beta(489,32)$, $\beta(489,280)$	$\beta(594,199)$, $\beta(594,175)$	$\beta(620,262)$, $\beta(620,149)$

Table 5.13.: Approximate composition operators for precision and recall

Collaboration pattern	**TP**	**FP**	**FN**
‘OR’ Collaboration	Min	Min	Max
‘AND’ Collaboration	Max	Max	Min

of the composed system. The predicted QoS values /distributions compared with the actual QoS values/ distributions are shown for 'AND' collaboration in Table 5.14 and 'OR' collaboration in Table 5.15.

Table 5.14.: QoS of the 'AND' collaboration

Feed	**Predicted (PR, RC)**		**Actual (PR, RC)**	
	Means	**Distributions**	**Means**	**Distributions**
PL1	(75, 37)	$\beta(12,4)$, $\beta(12,20)$	(84, 39)	$\beta(11,2)$, $\beta(11,17)$
PL2	(98, 81)	$\beta(64,1)$, $\beta(64,15)$	(100, 76)	$\beta(63,0.1)$, $\beta(63,19)$
PL3	(95, 77)	$\beta(60,3)$, $\beta(60,17)$	(98, 73)	$\beta(54,1)$, $\beta(54,19)$
PL4	(85, 60)	$\beta(40,7)$, $\beta(40,26)$	(92, 54)	$\beta(37,3)$, $\beta(37,31)$
PL5	(100, 64)	$\beta(11,0.1)$, $\beta(11,6)$	(100, 68)	$\beta(11,0.1)$, $\beta(11,5)$
PL6	(100, 63)	$\beta(29,0.1)$, $\beta(29,17)$	(96, 52)	$\beta(25,1)$, $\beta(25,23)$
PL7	(94, 36)	$\beta(17,1)$, $\beta(17,30)$	(100, 47)	$\beta(23,0.1)$, $\beta(23,25)$
PL*	(93, 64)	$\beta(234,16)$, $\beta(234,130)$	(96, 61)	$\beta(224,7)$, $\beta(224,139)$
SP1	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,2)$	(100, 20)	$\beta(1,0.1)$, $\beta(1,4)$
SP2	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,2)$	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,3)$
SP3	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,2)$	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,2)$
SP4	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,2)$	(NaN, NaN)	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$
SP5	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,2)$	(NaN, NaN)	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$
SP*	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,10)$	(100, 10)	$\beta(1,0.1)$, $\beta(1,9)$
ED1	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,6)$	(50, 14)	$\beta(1,1)$, $\beta(1,6)$
ED2	(NaN, NaN)	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,1)$
ED3	(NaN, NaN)	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$	(NaN, NaN)	$\beta(0.1,0.1)$, $\beta(0.1,0.1)$
ED4	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,1)$	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,1)$
ED5	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,3)$	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,3)$
ED*	(NaN, 0)	$\beta(0.1,0.1)$, $\beta(0.1,10)$	(50, 8)	$\beta(1,1)$, $\beta(1,11)$
ALL	(93, 60)	$\beta(234,16)$, $\beta(234,150)$	(96, 58)	$\beta(226,8)$, $\beta(226,159)$

Table 5.15.: QoS of the 'OR' collaboration

Feed	Predicted (PR, RC)		Actual (PR, RC)	
	Means	Distributions	Means	Distributions
PL1	(45, 68)	$\beta(22, 26)$, $\beta(22, 10)$	(44, 89)	$\beta(25, 31)$, $\beta(25, 3)$
PL2	(95, 87)	$\beta(69, 3)$, $\beta(69, 10)$	(100, 98)	$\beta(81, 0.1)$, $\beta(81, 1)$
PL3	(93, 88)	$\beta(68, 5)$, $\beta(68, 9)$	(92, 95)	$\beta(70, 6)$, $\beta(70, 3)$
PL4	(81, 89)	$\beta(59, 13)$, $\beta(59, 7)$	(85, 94)	$\beta(64, 11)$, $\beta(64, 4)$
PL5	(53, 82)	$\beta(14, 12)$, $\beta(14, 3)$	(42, 87)	$\beta(14, 19)$, $\beta(14, 2)$
PL6	(71, 82)	$\beta(38, 15)$, $\beta(38, 8)$	(68, 85)	$\beta(41, 19)$, $\beta(41, 7)$
PL7	(78, 82)	$\beta(39, 11)$, $\beta(39, 8)$	(67, 89)	$\beta(43, 21)$, $\beta(43, 5)$
PL*	(77, 82)	$\beta(299, 85)$, $\beta(299, 65)$	(75, 93)	$\beta(338, 107)$, $\beta(338, 25)$
SP1	(8, 50)	$\beta(1, 11)$, $\beta(1, 1)$	(17, 60)	$\beta(3, 14)$, $\beta(3, 2)$
SP2	(16, 50)	$\beta(1, 5)$, $\beta(1, 1)$	(0, 0)	$\beta(0.1, 4)$, $\beta(0.1, 3)$
SP3	(8, 50)	$\beta(1, 11)$, $\beta(1, 1)$	(14, 50)	$\beta(1, 6)$, $\beta(1, 1)$
SP4	(12, 50)	$\beta(1, 7)$, $\beta(1, 1)$	(0, NaN)	$\beta(0.1, 11)$, $\beta(0.1, 0.1)$
SP5	(11, 100)	$\beta(2, 16)$, $\beta(2, 0.1)$	(0, NaN)	$\beta(0.1, 17)$, $\beta(0.1, 0.1)$
SP*	(6, 30)	$\beta(3, 43)$, $\beta(3, 7)$	(7, 40)	$\beta(4, 52)$, $\beta(4, 6)$
ED1	(9, 16)	$\beta(1, 10)$, $\beta(1, 5)$	(30, 57)	$\beta(4, 9)$, $\beta(4, 3)$
ED2	(0, NaN)	$\beta(0.1, 2)$, $\beta(0.1, 0.1)$	(16, 100)	$\beta(1, 5)$, $\beta(1, 0.1)$
ED3	(NaN, NaN)	$\beta(0.1, 0.1)$, $\beta(0.1, 0.1)$	(NaN, NaN)	$\beta(0.1, 0.1)$, $\beta(0.1, 0.1)$
ED4	(0, 0)	$\beta(0.1, 4)$, $\beta(0.1, 1)$	(0, 0)	$\beta(0.1, 11)$, $\beta(0.1, 1)$
ED5	(0, 0)	$\beta(0.1, 8)$, $\beta(0.1, 3)$	(10, 33)	$\beta(1, 9)$, $\beta(1, 2)$
ED*	(4, 10)	$\beta(1, 22)$, $\beta(1, 9)$	(15, 50)	$\beta(6, 34)$, $\beta(6, 6)$
ALL	(69, 78)	$\beta(303, 134)$, $\beta(303, 81)$	(64, 90)	$\beta(348, 193)$, $\beta(348, 37)$

In Tables 5.14, and 5.15, the final row (corresponding to 'ALL' feed) represents the predictions and actual QoS values/ distributions, without conisdering the context. Whereas, the rows 'PL*', 'SP*', 'ED*', represent the context specific predictions and the corresponding actual values / distributions of QoS.

In Table 5.14, we can observe that after using the 'AND' operator, there is not enough data to predict the QoS for 'SP*' and 'ED*' feeds. However, even the context-independent predictor (corresponding to the 'ALL' feed) has failed to predict the QoS for 'SP*' and 'ED*' domains as the predictor over-estimate the QoS of these feeds. We conclude that in this type of applications, the ContextTrust model requires a significantly large amount of data from each domain (until adequate number of positives present), and if the data is not adequate to performs predictions, the predictions will not be accurate. In contrast, 'PL*' feeds has enough data to more accurately predict the QoS in that domain.

In Table 5.14, we can observe that after using the 'OR' operator, the context-dependent predictors ('PL*', 'SP*', and 'ED*' feeds) have provided accurate predictions for the corresponding feeds than the context-independent predictors (corresponding to 'ALL' feed). The difference between the use of 'OR' operator and the 'AND' operator is that more positives are predicted by the collaboration of classifiers when interacting with 'OR' mode. Therefore, the predictors have enough data to perform accurate predictions. To demonstrate accuracy of the 'ContextTrust' predictions compared to the prevalent method, we have plotted the cumulative distributions of the precision for the data from political domain (Figure 5.11), sports Domain (Figure 5.12) and education Domain (Figure 5.13). The figures show that the ContextTrust approach performs more accurate predictions than the prevalent approach that do not consider the context in QoS evaluation.

The application of the ContextTrust model can be generalized to domains other than the three case studies mentioned in this Chapter. 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 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 have used can be applied to large-scale Bayesian networks [93], we expect our model would perform equally well in complex systems as well.

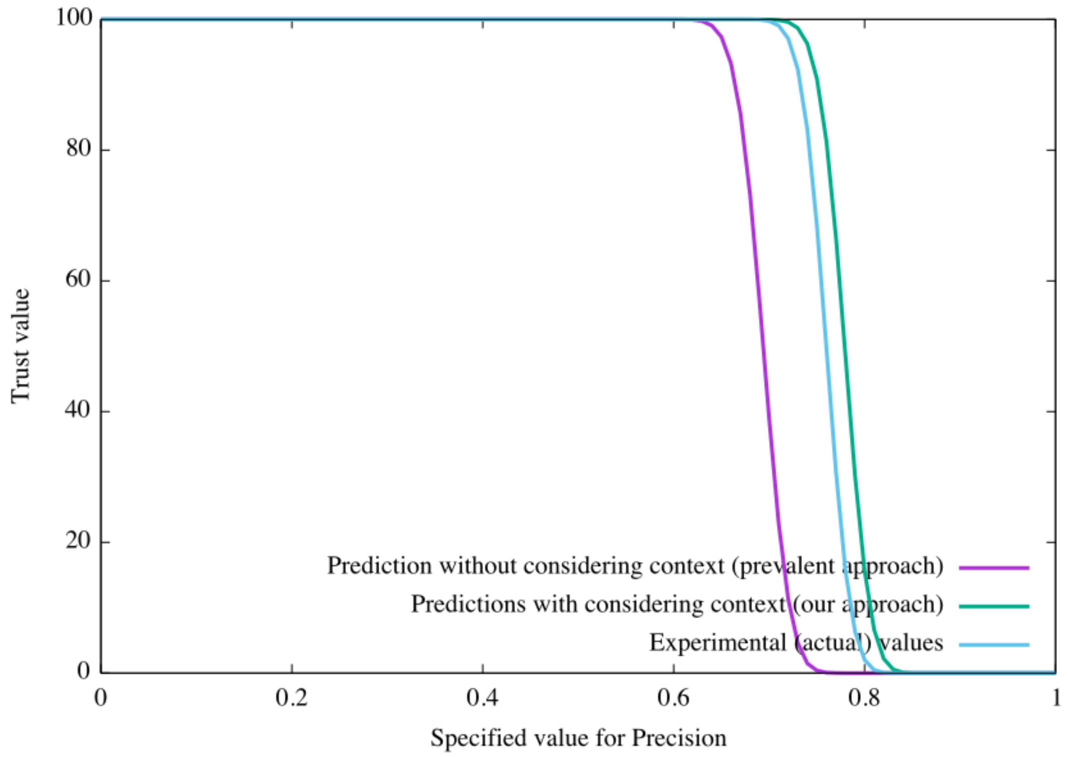


Figure 5.11.: A graph showing the trust of the precision for the data from political domain for different choice of specification values.

We obtained more accurate predictions about the QoS and the trust of systems by following the four phases enforced in ContextTrust model, which explicitly consider the context-QoS and the context-context dependencies. This would require additional cost at the very early phases of the software lifecycle. However, as the predictions provided by the proposed 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.

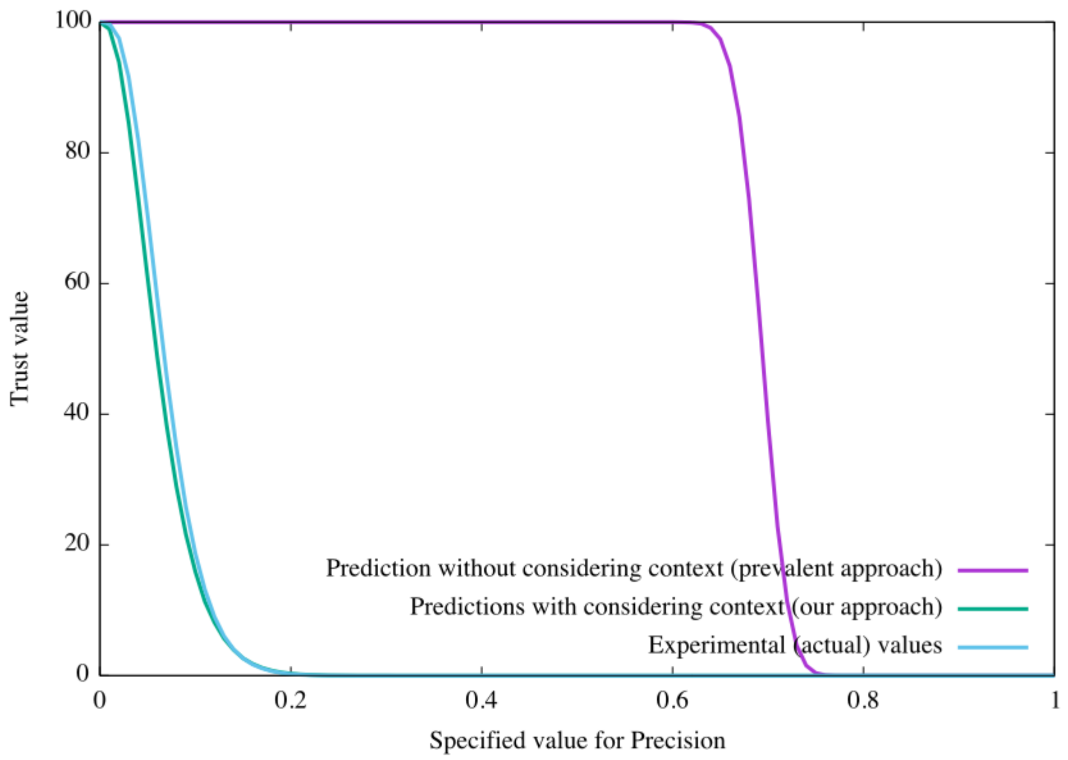


Figure 5.12.: A graph showing the trust of the precision for the data from sports domain for different choice of specification values.

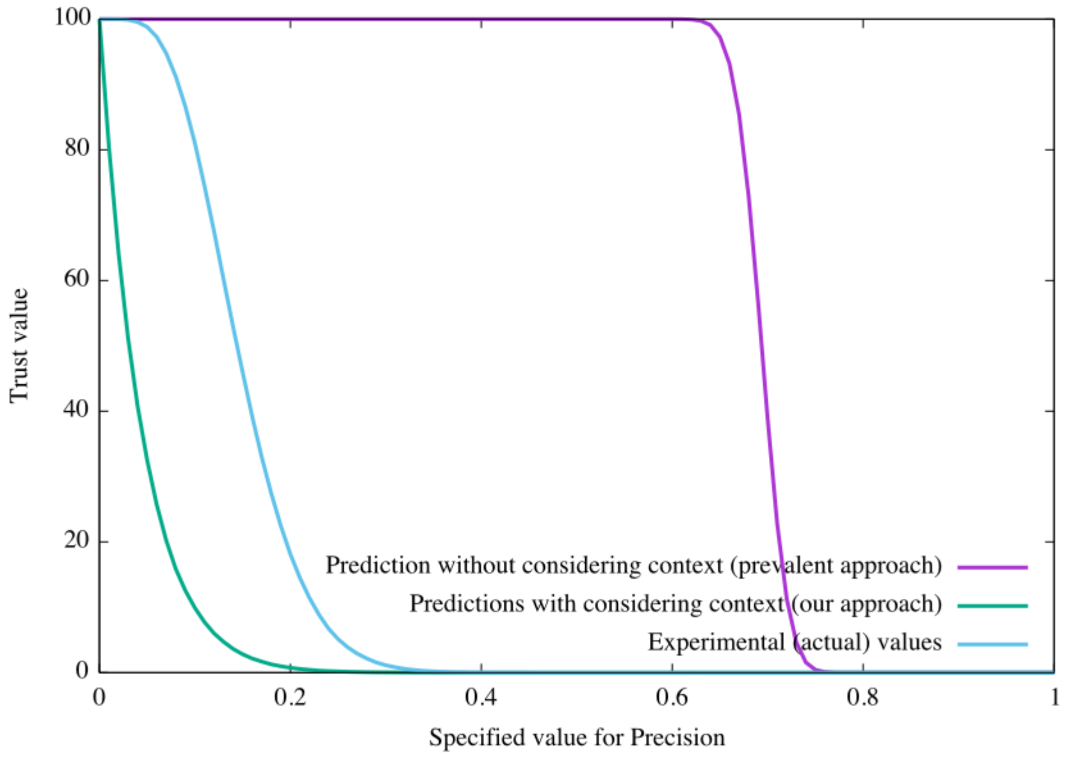


Figure 5.13.: A graph showing the trust of the precision for the data from education domain for different choice of specification values.

6 MORE APPLICATIONS OF THE QOS AND TRUST PREDICTION FRAMEWORK

The ContextTrust model is applied on real life case studies to predict QoS and Trust of composed systems at the design phase of the system in Chapter 5. The content in this chapter is an extension of our previous publications [20, 21]. In this Chapter, we discuss how we can extend the model to cater three other applications as listed below.

1. Selection of an optimum set of service for a composed system (OptimumTrust model)
2. Adaptation of the composed system based on QoS and Trust changes (AdaptTrust model)
3. QoS and Trust evaluations of individual services that are reused in existing composed systems (ReuseTrust model)

We present three models that are developed based on the ContextTrust model to provide solutions to applications listed above. These models are discussed in detail as separate sections of this chapter along with real life case studies for validation.

6.1 Optimum Service Selection Model – OptimumTrust

An important outcome that we can infer with the help of the 'ContextTrust' 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.

6.1.1 Optimization Algorithm

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

Let there be 'n' service groups $G_1, G_2, G_3, \ldots, G_n$, in which the services in a same group provide the same functionality, and for each service group G_i, there are 'm' number of service implementations, $S_{i1}, S_{i2}, S_{i3}, \ldots, S_{im}$. Each service (S_{ij}) has 'p' number of QoS parameters, $q_{ij}^1(C), q_{ij}^2(C), q_{ij}^3(C), ..., q_{ij}^p(C)$. 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-integer programing approach.

The binary variable X_{ij} is used to indicate whether the S_{ij} service is selected or not.

$$X_{ij} = \begin{cases} 1- & if \quad the \quad service \quad S_{ij} \quad is \quad selected \\ 0- & otherwise \end{cases} \tag{6.1}$$

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

$$\sum_{i=1}^{m} x_{ij} = 1 \quad for \quad all \quad j = 1 \ldots n \tag{6.2}$$

We have used the notation X to represent the set of X_{ij} for all i, j values. Each instance of the X that satisfies the equation 6.2 represents a candidate composition.

Furthermore, our target composite service should satisfy the constraints enforced on each quality value. Assuming the user requirement for the k^{th} QoS property is that its value should exceed $Q^k(C)$ with $T^k(C)$ trust, feasibility constraints can be modeled using the following inequality equations.

$$OP^k_{i=1\ldots n} \sum_{i=1}^{m} x_{ij} q^k_{ij}(C) \geq_{T^k(C)} Q^k(C) \quad for \quad all \quad k = 1 \ldots p \tag{6.3}$$

Here the OP^k represents the composition operator corresponding to the k^{th} QoS property and the operator $\geq_{T^k(C)}$ represents the inequality should holds with at least $T_k(C)$ probability for a given feasible composition. Here, we assume that a higher QoS value for q^k_{ij} is desirable. If lower QoS values are desirable, then negative values should be assigned to q^k_{ij} to make sure to use the same inequality equation.

The left hand side of the equation 6.3 can be evaluated using ContextTrust model discussed in Chapter 4. It provides the probability distributions of overall QoS values for a composition of some selected candidate services. Then we would use the cumulative probability distributions of QoS values 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 'k'th QoS value that satisfies the trust requirement is denoted as $Q^k_{T^k(C)}(X)$, then the equation 6.3 can be rewritten as following.

$$Q^k_{T^k(C)}(X) \geq Q^k(C) \quad for \quad all \quad k = 1 \ldots p \tag{6.4}$$

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.

$$F = \sum_{k=1}^{p} W^k Q^k_{T^k(C)}(X) \tag{6.5}$$

Zeng et al. [55] 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, where as 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.

6.1.2 Cross Entropy (CE) Algorithm

We have used an optimization algorithm based on the cross-entropy method [16,94] 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 [16] is summarized in the following steps. The parameters ρ, d, α 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 groups. 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 ρ between 0 and 1, where last ρN samples are

considered relatively optimum samples. In the next steps, we would adjusts the P_{ij} values favoring these ρN samples.

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

$$C_{ij} = \frac{\sum_{r=1}^{N} I_{F_R > \gamma} x_{ij}}{\sum_{r=1}^{N} I_{F_R > \gamma}} \tag{6.6}$$

   Where, $\gamma = F_{(1-\rho)N}$ and

$$I_{F_r > \gamma} = \begin{Bmatrix} 1- & if \quad F_r > \gamma \quad for \quad sample \quad `r' \\ 0- & otherwise \end{Bmatrix} \tag{6.7}$$

5. Update the P_{ij} values for the next iteration for each service S_{ij}.

$$P_{ij} = \alpha C_{ij} + (1-\alpha) P_{ij} \tag{6.8}$$

6. If the γ values corresponding to the last d number of iterations are equal, then we can decide the solution is converged and the γ 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 optimality of 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 '$i-1$', 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 $i-1$ 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 $(L(S_{ij}|s_1,\ldots,s_{i-1}))$ using the following expression.

$$L(S_{ij}|s_1,\ldots,s_{i-1}) \quad \propto \quad 1+\frac{\Delta F_{ij|s_1,\ldots,s_{i-1}}}{\left|\Delta F_{ij|s_1,\ldots,s_{i-1}}\right|+|F_{ij}|} \tag{6.9}$$

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. $\Delta F_{ij|s_1,\ldots,s_{i-1}}$ is the difference of the objective function from the service S_{ij} made due to the associations with the services $s_1,\ldots,s_{i-1}$. In this equation, if there are no associations between the already selected services and the service S_{ij}, ΔF becomes 0 and the likelihood would be proportional to 1. If there is a positive association, then the ΔF would become a positive fraction, and the likelihood would be proportional to L, $1 < L < 2$. If there is a negative association, then the ΔF 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 $P(S_{ij}|s_1,\ldots,s_{i-1})$) can be evaluated using Bayes theorem, in which P_{ij} is the prior probability.

$$P(S_{ij}|s_1,\ldots,s_{i-1}) \quad \propto \quad L(S_{ij}|s_1,\ldots,s_{i-1})P_{ij} \tag{6.10}$$

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 i is temporary updated to P'_{ij} for the purpose of the sampling from that service group.

$$P'_{ij} = \beta_t P(S_{ij}|s_1,\ldots,s_{i-1}) + (1-\beta_t)P_{ij} \tag{6.11}$$

Here the β_t is a parameter that would diminish as the iteration number(t) reaches higher values. In our experiments, we have used $\beta_t = \frac{\beta}{t}$ for $t < k$ 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 6.6). Then, after we select the prominent ρN 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.

6.1.3 Ant Colony Optimization (ACO) Algorithm

ACO algorithm is based on the behaviour of ants, when finding the optimum path from their colony to a food source [95]. When ants travel from source to the destination through several nodes, they lay down pheromone in the trail. Subsequent ants tend to follow the highest concentration of pheromone in deciding the next node of the path. As the pheromone evaporate with time, shorter paths tend to have high concentration of pheromone. With time, this leads the ants to find the optimum path. ACO simulate this behavior by evaluating probabilities of tendency to travel between pair of nodes. These probabilities are re-evaluated based on the cost of the selected path. As in our application, the optimum solution tend to have positively associated services, we can use the ACO algorithm to capture such associations in solving the optimization problem. The steps of the algorithm that is applied in optimum service selection problem are listed below.

1. If the service j is selected (in the service group i), then the probability (P_{ijk}) that the service k is getting selected (in the service group $i+1$) is evaluated using the following equation.

$$P_{ijk} = \frac{(\tau_{ijk}^{\alpha})(\eta_{ijk}^{\beta})}{\sum_{k=0}^{k=m}(\tau_{ijk}^{\alpha})(\eta_{ijk}^{\beta})} \tag{6.12}$$

   Here α, and β are tuning parameters. τ_{ijk} is an evolving probability value (corresponding to the pheromone concentration) that is initialized uniformly. η_{ijk} is the prior knowledge of the associations. For benchmark purposes, when we do not consider associations between services, we initialize the η_{ijk} values uniformly. When the associations are considered (refer as ACO + Heuristics in the graphs in Sub-section 6.1.4), we initialize η_{ijk} as following equation.

$$\eta_{ijk} \quad \propto \quad 1 + \frac{\Delta F_{(i+1)k|s_{1j}}}{\left|\Delta F_{(i+1)k|s_{ij}}\right| + \left|F_{(i+1)k}\right|} \tag{6.13}$$

   Similar to Equation 6.9, this equation provides a η_{ijk} value greater than 1, if the services are positively correlated, less than 1, if the services are negatively correlated, and equal to 1, if there are no association between services.

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 groups. In each sample, if the service j is selected in the group i, we follow P_{ijk} to select the service k from the $i+1$ group.

3. For each sample r (from the N samples), evaluate the objective function Ob_r.

4. After the samples are drawn, the P_{ijk} values are updated as in following equation.

$$\tau_{ijk} = (1-\rho)\tau_{ijk} + \sum_{r=1}^{N} \Delta\tau_{ijk}^{r} \tag{6.14}$$

Here $\Delta\tau^r_{ijk}$ is evaluated for each sample r as in following equation.

$$\Delta\tau_{ijk} = \begin{Bmatrix} \frac{Ob_r}{Q}- & If \quad the \quad sample \quad r \quad contain \quad j \quad and \quad k \quad services \\ 0- & otherwise \end{Bmatrix} \tag{6.15}$$

5. Continue the Step 2 for the iteration $t + 1$ unless the in the last d number of iterations, κ portions of the samples have the same highest objective function.

6.1.4 Simulation Study of the Optimization Algorithm

In order to test the validity of the algorithm with the proposed heuristic that capture the association between services, we applied the algorithms to a simulated data set. Simulations have been used in previous works [14–16] 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 algorithms 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). The default values for these parameters as shown in the Table 6.1. Here the U(X,Y) stands for random numbers that are sampled from uniform distribution between X and Y.

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 6.1.

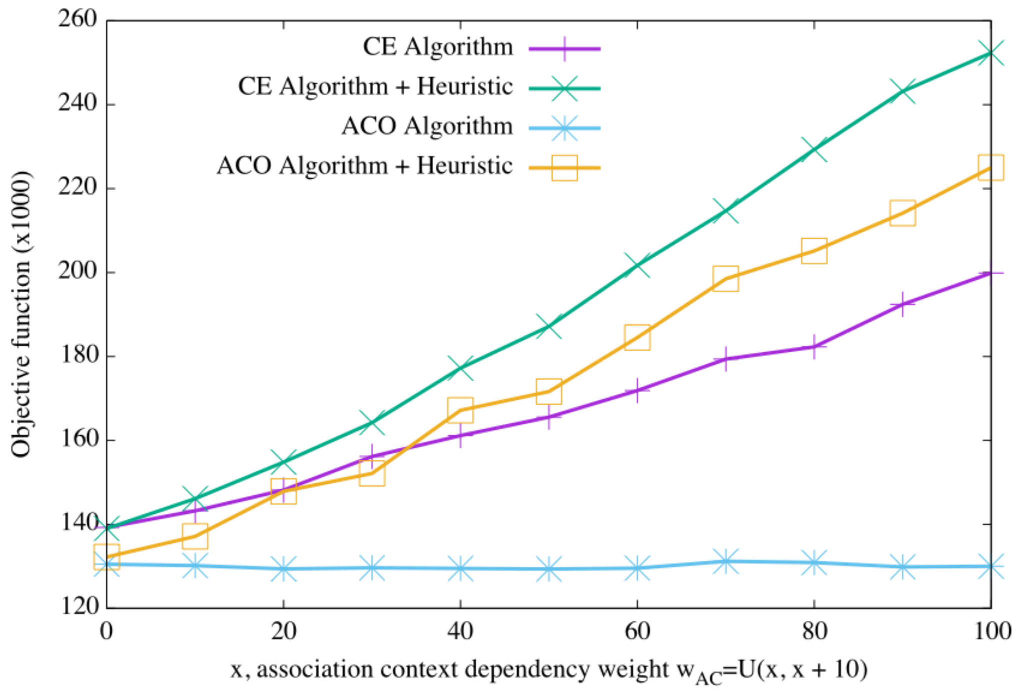


Figure 6.1.: Objective function vs the association context dependency weight

From Figure 6.1, It is clear that the CE algorithm outperforms the ACO algorithm for all the different association context dependency weights. It also shows that when the associativity context is prominent, our heuristic methods are effective in getting a comparatively higher objective values for both CE and ACO algorithms. Figure 6.2 shows that ACO algorithm takes only fewer iterations than the CE algorithm. Additionally, the algorithms have converged to an optimum value in lesser number of iterations, when the heuristic about the associations are used. However,

Table 6.1.: Default values for the simulation and algorithm tuning parameters

Simulation Parameters	**Value / Random Value Distribution**
n	1000
m	20
c	5
q	5
$\|AC\|$	1
C	$U(0,1)$
w_C	$U(10,20)$
w_{AC}	$\pm U(10,20)$
CE Algorithm Tunning Parameters	**Value**
N	20
ρ	0.2
d	10
α	0.1
β	0.4
k	30
ACO Algorithm Tunning Parameters	**Value**
N	20
ρ	0.1
d	10
α	1
β	5
κ	0.9
Q	150000
Objective function/Feasibility Constraints Parameters	**Value**
$W^1, \ldots, W^q$	$15 \times n \times c \times q$
$Q^1, \ldots, Q^q$	1
$T^1, \ldots, T^q$	100%

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 6.3. Even with lesser number of iterations, ACO algorithm have taken significantly more time, as it requires calculation of probabilities for each pair of services in adjacent groups in each iterations. If the requirement is to evaluate 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. If the requirement is to evaluate the compositions at runtime, we would need the algorithm to be fast while providing the optimum solution. The graph in Figure 6.4 highlights that the CE + Heuristic algorithm performs equally fast compared with the CE algorithm. Therefore, CE algorithm with proposed heuristics is capable of providing comparatively higher optimum solutions for systems with high association context dependencies taking comparatively lesser time.

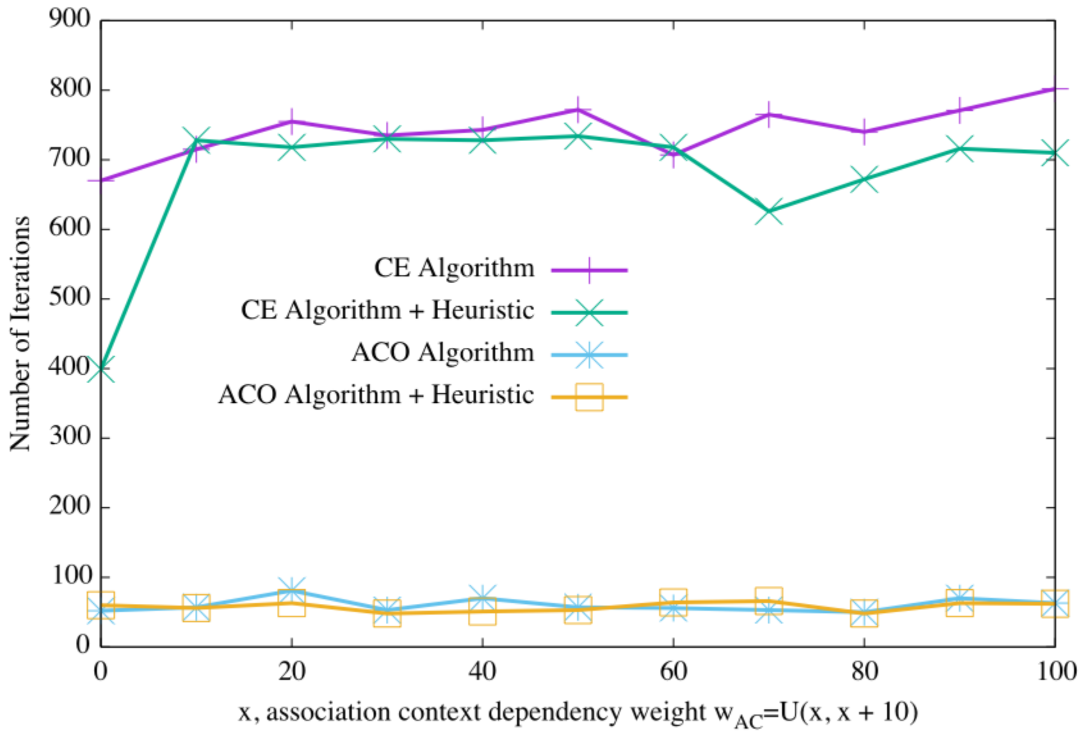


Figure 6.2.: Algorithm iteration vs the association context dependency weight

Similarly, we have performed experiments by changing the number of service groups (Figure 6.5) and number of services for a service group (Figure 6.6) with the same association context dependencies that is generated at $U(100, 110)$ and eval-

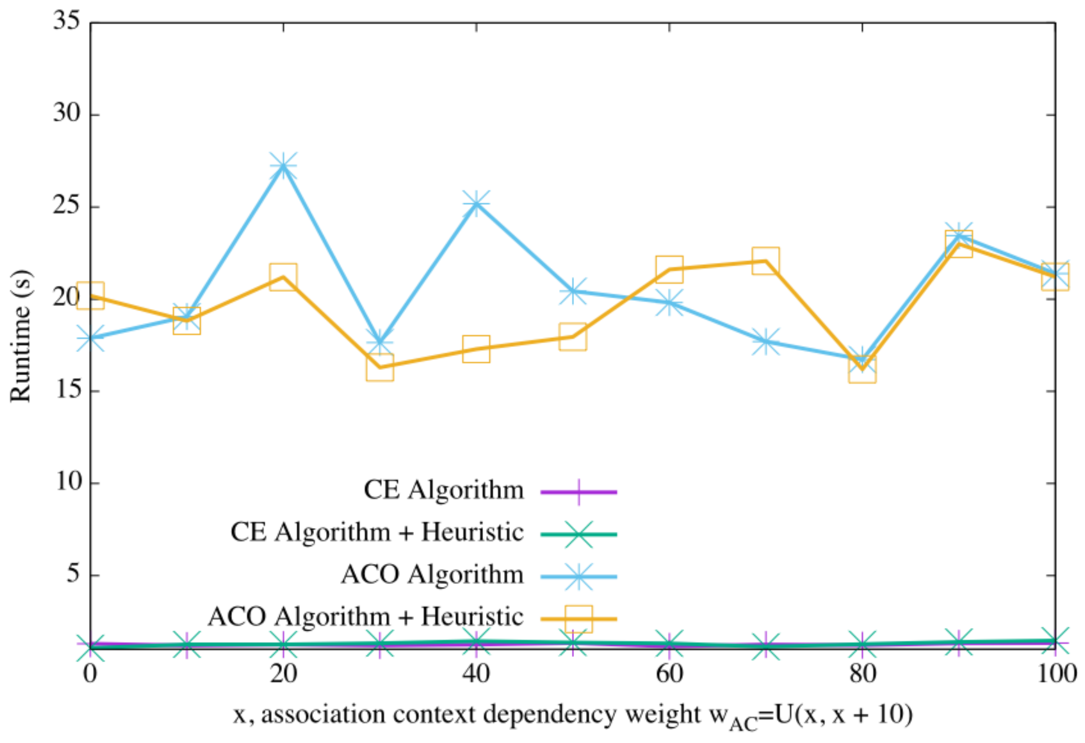


Figure 6.3.: Algorithm runtime vs the association context dependency weight

uated the effectiveness of the algorithm. The graphs in Figure 6.5, and Figure 6.6 show that the both the heuristic-based algorithms find optimum solutions than the non-heuristic algorithms consistently.

In Figure 6.5, the value of the 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 for both CE and ACO algorithms have captured this associations more effectively and provide 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. With the availability of more choices, there will be many feasible

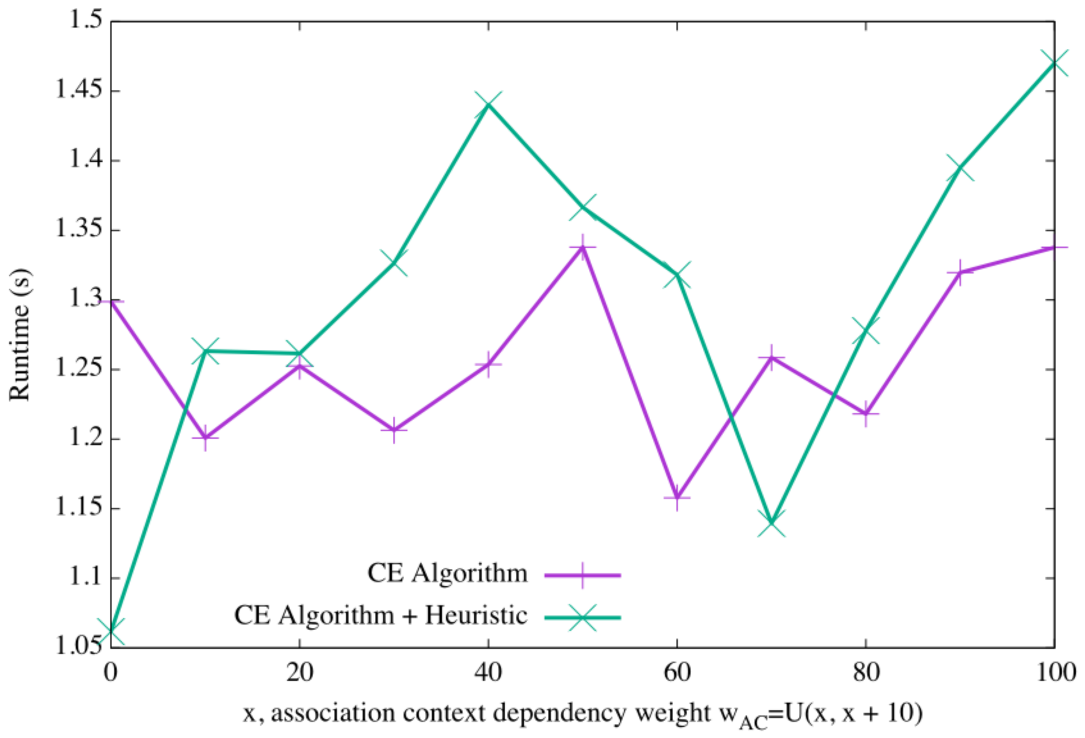


Figure 6.4.: Runtime vs the association context dependency weight for CE, CE+Heuristics algorithms

compositions with optimum QoS and trust values although the services within the composition do not necessarily be associated with each other. Figure 6.6 shows that the proposed heuristic-based algorithms performed better than the traditional algorithms. However, with the increase of the number of services per group, the traditional algorithms come closer to heuristic-based algorithms. After analyzing optimum solutions from each of these algorithms, we conclude as the number of services increases, optimum compositions tends to happen with and without associations between the services. The heuristics-based algorithms tend to capture the compositions with associations between services, and traditional algorithms tend to capture compositions without associations between services. Therefore, with the increase of the services in a service group, traditional algorithms come closer to the optimality of the heuristic based algorithms.

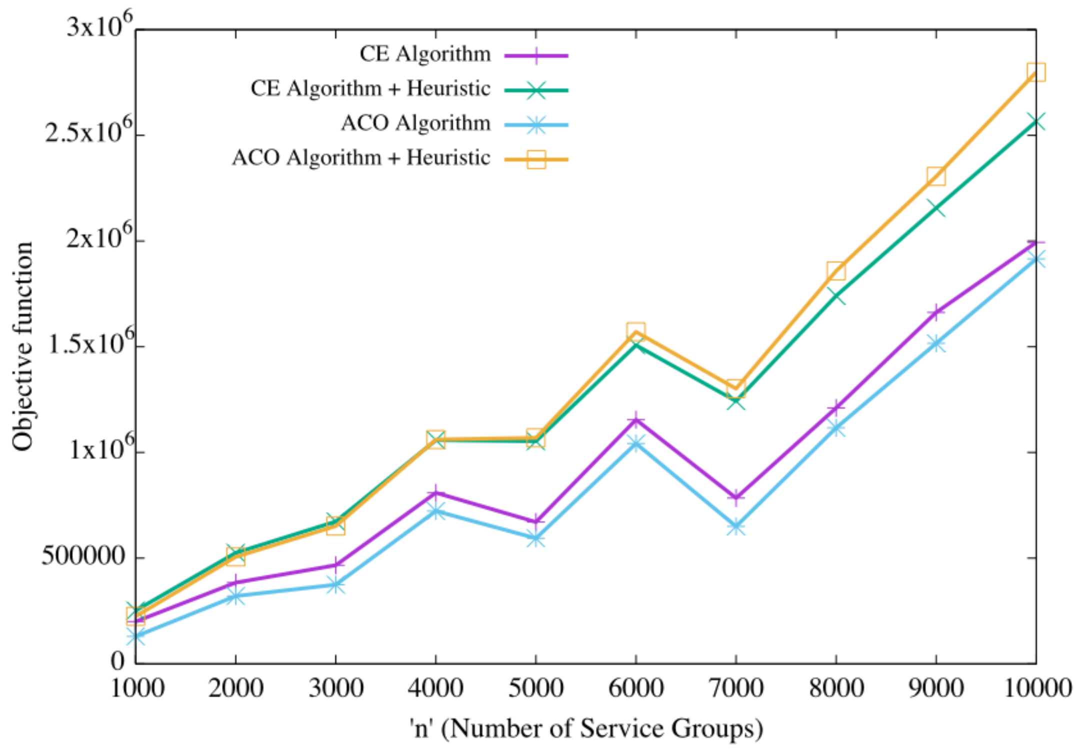


Figure 6.5.: Objective function vs number of service groups

6.1.5 Case Study – A Travel Planning System

We have applied our optimum service selection algorithms to a travel planning system that we used as a case study in Section 5.2. The dataset of the system (that is the service groups and the corresponding candidate services) is listed in the Table 3.6.

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 proprieties. For example, if the security requirement for the system is that it should support authentication/ autho-

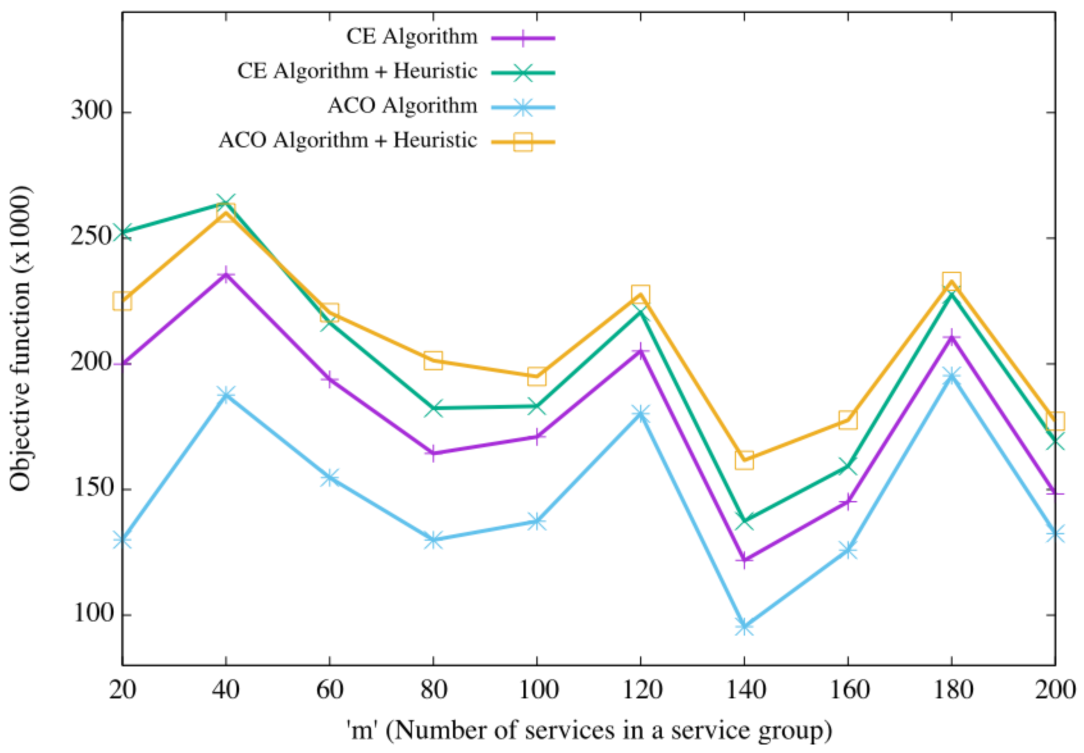


Figure 6.6.: Objective function vs the number of services in a service group

rization, 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 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.

All the algorithms (CE, CE+Heuristics, ACO, and ACO+Heuristics) provided the same optimum solution for each different contexts in the dataset. As the number of service groups and services per groups are low in this situation, we could verify that they are the global optimum solutions by validating the solution with brute-force technique. Table 6.2 shows the overall QoS values for the optimum composition for two contexts using both the proposed algorithms and the actual executions of compositions. (In Table 6.2, only the 'Rst' is mentioned as the 'Auth', 'Nrep', and 'Conf' Qos properties can be implied from the context.)

Table 6.2.: The overall QoS values for optimum composition for different contexts

Context	**Predicted Rst**	**Actual Rst**
$Usr = true$, $Dist \approx 200miles$	1666ms (DR1, TR1, HO3, WE2, CR1)	1837ms (DR1, TR1, HO3, WE2, CR1)
$Usr = true$, $Dist \approx 3000miles$	4106ms (DR2, TR1, HO3, WE2, CR1)	4025ms (DR2, TR1, HO3, WE2, CR1)
$Sign = true$, $Dist \approx 200miles$	2478ms (DR1, TR1, HO3, WE2, CR1)	2713ms (DR1, TR1, HO3, WE2, CR1)
$Sign = true$, $Dist \approx 3000miles$	6504ms (DR2, TR1, HO3, WE2, CR1)	6990ms (DR2, TR1, HO3, WE2, CR1)
$Enc = true$, $Dist \approx 200miles$	2797ms (DR1, TR1, HO3, WE2, CR1)	3013ms (DR1, TR1, HO3, WE2, CR1)
$Enc = true$, $Dist \approx 3000miles$	6537ms (DR2, TR1, HO3, WE2, CR1)	7069ms (DR2, TR1, HO3, WE2, CR1)

From the results shown in Table 6.2, 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 where as MapQuest Direction Service (DR2) sends a comparatively smaller output. For example, the messages sizes for the direction between Indianapolis and Chicago (around 200miles distance) are 32kb (DR1), and 14kb (DR2), where as the message sizes for the direction between New York and San Francisco (around 3000miles distance) are 245kb (DR1), and 31kb (DR2). The reason for this is 'DR1' provides more alternative routes as distance increases, and 'DR2' only provides few optimum routs. 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 6.3. 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 travellings, and DR1, TR1, HO3, WE2, and CR1 services for compositions that deal with low distance travellings. Furthermore, the relative absolute errors of the prevalent approach and the proposed approach in predicting ‘Rst’ for the optimum compositions is compared in Table 6.4. As the ’Rst’ strongly depends on all the context parameters, it is clear the consideration of such dependencies significantly improve the prediction error.

Table 6.3.: QoS predictions of optimum compositions without considering the context

Candidate Composition	**Predictions of ‘Rst’ without considering the context**
DR1, TR1, HO3, WE2, CR1	4133.3ms
DR2, TR1, HO3, WE2, CR1	4036.8ms

6.2 QoS and Trust Based Adaptation Model – AdaptTrust

The ContextTrust model proposed in Chapter 4 helps the developers to evaluate QoS and Trust values of composed systems during the early phases of the system development life-cycle. However, in IoT and CPS domains, the context changes frequently and hence, it is a more difficult challenge to develop self-adaptive sys-

Table 6.4.: QoS predictions of the optimum service composition

Approach	**Relative absolute error of predictions of 'Rst'**
Bayesian network based approach (Considering the context)	0.204
Prevalent approach (Without considering the context)	0.949

tems to continuously operate while satisfying user QoS and Trust requirements. In this section, we address these issues by proposing an adaptation model (named as AdaptTrust) that developers can rely on to develop self-adaptive, trustworthy, and distributed systems found in the IoT and CPS domains. The AdaptTrust model extends the ContextTrust model to operate at runtime with heuristic-based fast inferences techniques.

The AdaptTrust model is capable of feeding data to adaptation services about changes on QoS, Trust, and Context. Additionally, the model notifies adaptation services with the necessary information about when the adaptation has to trigger to keep the QoS and Trust values of the system within a user satisfactory level.

In the AdaptTrust model, we expect the developers to complete the four stages of the ContextTrust model at the design time of the system. The reason is that the structure of the context-QoS dependencies of services are expected to be fixed over time, therefore, the Bayesian networks corresponding to the participating services do not drastically change. After the Bayesian networks are ready at the design time, inferences are made at runtime with the changing contexts.

The following subsections discuss the components of the AdaptTrust model and the improvements done to the ContextTrust model to make it adaptation ready at runtime. We use the indoor tracking system (introduced in Section 5.1) with adaptation capabilities as the case study to validate the proposed model.

6.2.1 Application of the ContextTrust Model at Runtime

We use the ContextTrust model at runtime to capture the changes in QoS and trust values starting from individual services to composed systems. Furthermore, we discuss the types of causes for such changes, which we refer as 'triggers'.

Context Triggers: Triggers that cause the changes in the context properties of the Context-QoS dependency Bayesian networks are referred as the 'Context Triggers'. When Context Triggers occur, the Context-QoS dependency Bayesian networks of corresponding individual services and composed systems can be used to evaluate the corresponding change in QoS and Trust values. For example, in the Camera Tracking Service (Figure 4.3), the changes in the position of either the tracking object or the camera will change the distance and angle, which are context properties. The AdaptTrust model can capture such changes and calculate the corresponding tracking error and response time.

Replacement Triggers: Triggers that replace the components (corresponding to individual services) of the composed Context-QoS dependency Bayesian networks are referred as the 'Replacement Triggers'. When such triggers occur, the QoS and Trust values are re-evaluated for the newly composed Context-QoS dependency Bayesian network. Replacement Triggers can happen due to the availability of new services, malfunctioning of existing services, or as a result of 'Context Triggers'. An example of a Replacement Trigger caused by a Context Trigger is, in the Camera Tracking Service (Figure 4.3), when the objects move out of the sight of a camera (due to movements of either the object or camera, which is a change in its Context), the camera may be replaced with another camera that has the object in its sight causing a Replacement Trigger.

Interaction Triggers: Triggers that replace the interaction patterns between the components (corresponding to individual services) of the composed Context-QoS dependency Bayesian networks are referred as the 'Interaction Triggers'. Such changes are due to the changes in branching, aggregating, and looping conditions of the composition. These conditions may also depend upon changes in the system context

('Context Triggers'). For example, an elastic load balancer service will increase or decrease the number of active servers according to the increase or decrease of the request load. As only the context triggers and the replacement triggers are applicable in our case study, we do not discuss further the adaptation process for interaction triggers.

6.2.2 Detection and Adaptation Services

Our AdaptTrust model keeps track of such triggers and reacts when QoS and Trust values are dropped below the user requirements. The high level architecture of the proposed AdaptTrust model (when applied to a composed system) is illustrated in Figure 6.7.

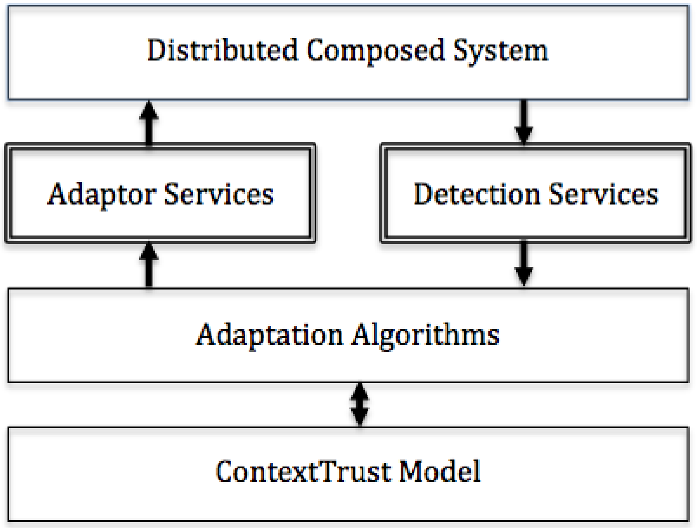


Figure 6.7.: The high level architecture of the AdaptTrust model

The model requires two types of services (listed below) to help its adaptation operations on the composed system.

1. **Detection Services**: These services detect the triggers and feed the data about the triggers to the model. An example of such a detection service is the Camera

Tracking Service (Figure 4.3) that can detect the position of the tracked object, which can be used to assess the changes in its own context (distance and angles).

2. **Adaptation Services**: These services trigger reactions (which we refer as 'adaptation triggers') and attempt to neutralize the effects of the triggers. An example of such a adaptation service is the camera Adaptation Service (which triggers pan and tilt operations on the camera to recover its context to obtain a high tracking accuracy). It is represented in the Bayesian network shown in Figure 6.8.

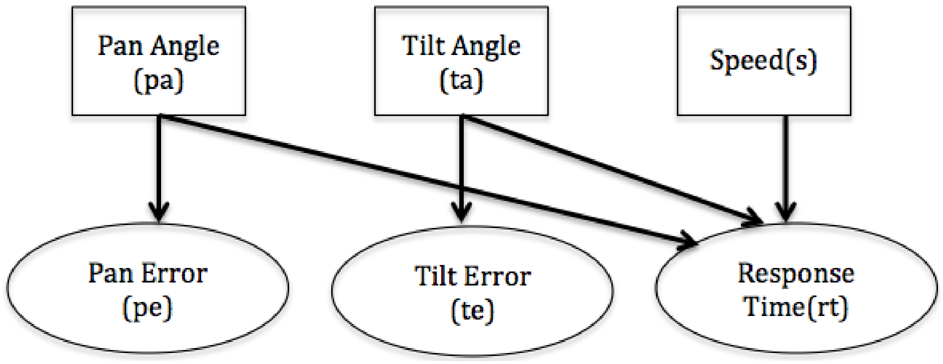


Figure 6.8.: Bayesian network of the camera adaptation service.

The operations of the Detection and Adaptation Services depend on the type of triggers incident on the system. The model uses these services to invoke the QoS and Trust Adaptation algorithms and maintain the QoS and Trust values of the system in the required range. These algorithms include the Context Adaptation Algorithm, which adapts the system due to Context Triggers, and the Replacement Adaptation Algorithm, which adapts the system due to Replacement Triggers. The Replacement Adaptation Algorithm listens for the Replacement Triggers, and when needed, invokes the appropriate Replacement Adaptation Services. For example, Replacement Adaptation service could implement OptimumTrust model discussed in Sub-section 6.1 to re-evaluate the optimum subset of services for a composed system at runtime. In contrast, the Context Adaptation Algorithm has several steps discussed in detail below.

6.2.3 Context Adaptation Algorithm

The Context Adaptation Algorithm recovers the QoS and Trust values of a service caused by the context triggers. The context adaptation is done with the help of the Context Detection Services, Context, and Replacement Adaptation Services. In AdaptTrust, we follow the following steps:

1. Achieve a continuous evaluation of the context by:
    (a) Measuring the context triggers that cause the context change.
    (b) Measuring the context directly using Detection Services.
2. Evaluate the deviation of QoS and Trust values from the required values by forward inferencing the Bayesian networks [96].
3. Evaluate the adjustments to the context that needs to achieve the required QoS and Trust values of the service by backward referencing [96].
4. Recover the desired context using the Adaptation Services.

6.2.3.1 Continuous evaluation of the context

For the context adaptation process, it is important to continuously monitor the context and detect its changes. The required frequency of the measurements will depend on the duration of the failure to satisfy QoS and Trust values that can be tolerated by the system users. In the case study, we have used 100ms to be the measurement interval, as it is adequately higher than the average request processing time (35ms).

If there is a context change, due to a known context triggering action, we measure the triggering action and assess the context change based on that measurement. Specially, when the AdaptTrust model generates adaptation triggers to recover the context, it knows the amount of context change it triggered. For example, in a Camera Tracking Service (Figure 4.3), an adaptation trigger can be achieved by executing the pan and tilt operations on the camera (using the Camera Adaptation Service in Figure 6.8). As the model executes the pan and tilt commands, it will be able to calculate the updated context parameters (such as distance and angles) using the triggered pan and tilt measurements without the need of measuring the context directly using a Context Detection Service.

Similarly, if there are Context Detection Services that directly measure the context changes (such as the Camera Tracking Service that track an object's position), we can use these services to continuously evaluate the context.

When both types of the measurements (triggering action measurement and the context change measurement) are available for a particular system, we can use them for a better assessment of the context using the Kalman filter [97] predictions. Here, we make the markov assumption [98] that the state of the context depends only on the immediate preceding context, in addition to the Gaussian assumption made about the distribution of the context values. With these assumptions, we map the context trigger to be the state-transition model [97] and the context detectors to be the observation model and evaluate the posterior distribution of the context iteratively. The noise distribution corresponding to both the state-transition model [97] and the observation model can be evaluated using QoS of the Bayesian network. For example, when evaluating the pan/tilt of the camera, the noise of the observation model can be evaluated using the Bayesian network of the Camera Tracking Service (Figure 4.3) the noise of the state-transition model using the Bayesian network of Camera Adaptation Service (Figure 6.8). The process of evaluation of the context and its relation to the Bayesian network that represents the context-QoS dependencies is illustrated in Figure 6.9.

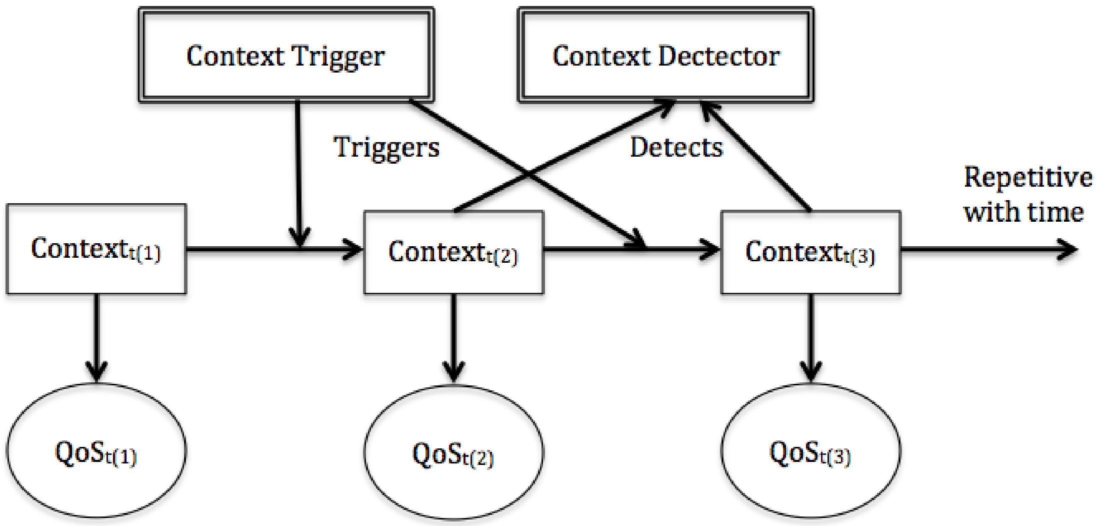


Figure 6.9.: Evolution of the context with time with the Markov assumption.

6.2.3.2 Evaluating the deviation of QoS and trust

After we evaluate the context using the Step 1, we use the Bayesian networks to infer the QoS and Trust values of the service. Such inferences can be done using forward sampling [58]. We generate samples from the Gaussian distributions with the mean and variations evaluated in Step 1 for contexts (which are independent variables), and generate the samples for QoS features (which are dependent variables). The resultant distribution of the samples corresponding to QoS are used to evaluate the Trust (as Trust equals to the percentile that satisfy the user requirements). If the QoS and the Trust values do not satisfy the user specified thresholds, we would continue to the next step of the algorithm described below.

6.2.3.3 Evaluating the adjustment of context

When the QoS and Trust values drop below the preferred thresholds, and if there are context Adaptation Services available, we evaluate the necessary adjustments that can be made to the context to recover the QoS and Trust. For example, in the Camera Tracking Service (Figure 4.3), if the response time(rt) and tracked error(e) observed have user preferred values when tracking an object on distance (d), we can predict the corresponding context values for the angle(a) using backward inference techniques. For that, we use rejection sampling [58]. However, rejection samples are very inefficient specially for large Bayesian networks with low probability of occurrences. By using Gibbs sampling combined with rejection sampling [99], we can speed up the inferences on high dimensional Bayesian networks by sampling one dimension at a time. However, since these types of inferences have to be run more frequently and are expected to provide results within real time constraints, we use two additional heuristics to speed up the inferences. These are:

1. **Cache the set of contexts**: Store a set of context values that provide the user preferred QoS and Trust values. Such context values can be found using experimental off-line execution of the system. This will be useful in situations where we can control all the context values using the context adaptation triggers or when context values have limited number of discreet values. For example in Camera Tracking Service (Figure 4.3), the resolution width and height can only take limited number of values. Therefore, we can cache resolution width and height values that provide the user preferred tracked errors and response time. This heuristic saves the overhead of having to sample each time we want to infer contexts. However, if there are some context values that we do not have the control of or that can have many continuous values, it is not practical to cache large amount of context values. In such cases, we are using the next heuristic to make fast inferences.

2. **Cache the samples**: Store the samples of the Bayesian network in a cache when the service operations satisfy the user QoS and Trust requirements. In a situation where context changed in a way that the QoS and Trust values no longer satisfy the user requirements, we can retrieve these samples from the most recent cache entries, and use them as initializations of the Gibbs sampling. The assumption used in this heuristic is that when the context is changed to deviate the QoS and Trust values from the preferred thresholds, the context does not

change drastically and only a few context parameters changes at a given time. For example, in a Camera Tracking Service (Figure 4.3), the angle (a) to the tracked object that provides the least tracking error, will only depend on the distance to the object regardless of the pan and tilt of the camera. Therefore, if we have cached the samples for different distance (d) values, we will be able to use the same set of samples when we need to infer for closer distances.

6.2.3.4 Recovering the context

After the preferred context values are calculated, the AdaptTrust model delegates the task of recovering the context to the Context Adaptation Service with the necessary data about the required changes of the context values. If the Context Adaptation Service cannot restore the context (as the required amount of change is out of their capability bounds), then the model initiate a replacement trigger and delegates the task to Replacement Adaptation Algorithm, discussed earlier.

6.2.4 Case Study – Adaptive Tracking System

We have used an indoor tracking system [100] as a case study to validate the proposed AdaptTrust model. This is an alteration of the case study we have used to validate the ContextTrust model in section 5.1. The altered version of the tracking System contains adaptation services and is referred as the adaptive Tracking System. This system is composed of four camera tracking services, two camera adaptation services, a fusion service and a fusion adaptation service. The physical setup of the adaptive tracking system is shown in Figure 6.10. A video of a sample experiment is available at [101]. The interaction patterns between the services for the system are shown in Figure 6.11.

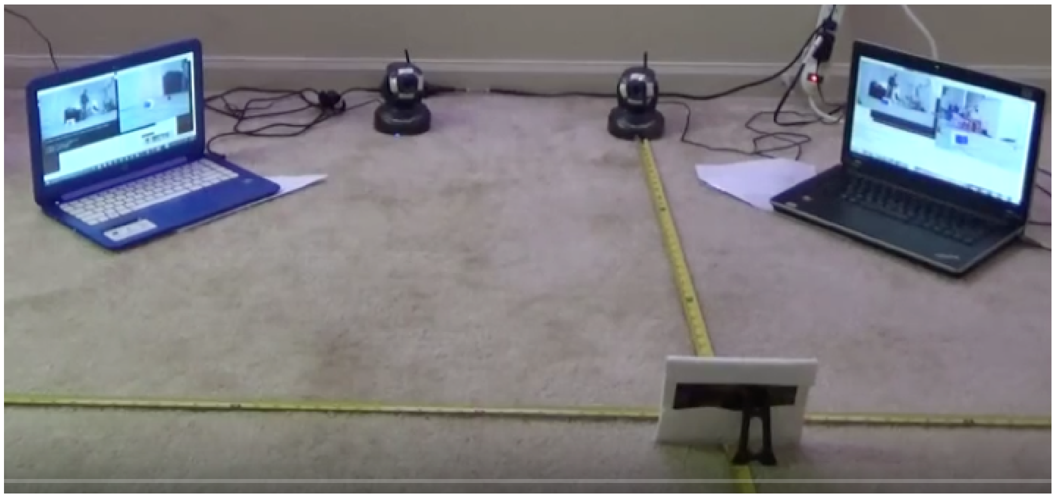
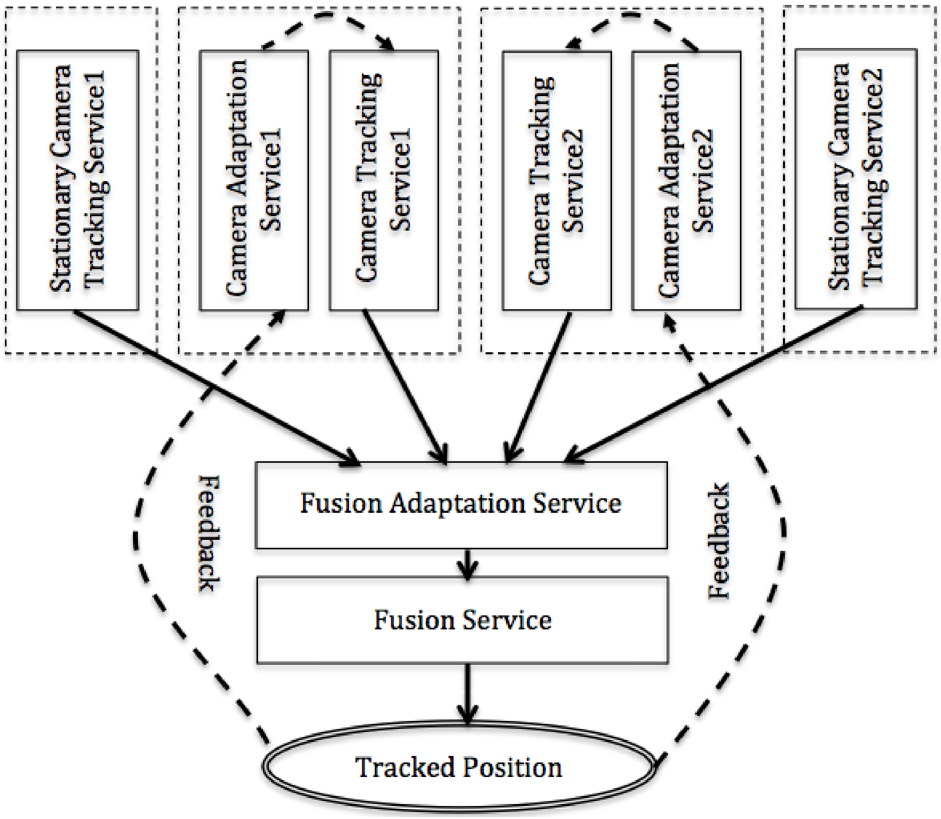


Figure 6.10.: Physical setup of the adaptive tracking system.


Stationary Camera Tracking Service1
Camera Adaptation Service1
Camera Tracking Service1
Camera Tracking Service2
Camera Adaptation Service2
Stationary Camera Tracking Service2
Feedback
Fusion Adaptation Service
Fusion Service
Feedback
Tracked Position
Figure 6.11.: Services and their interactions in adaptive tracking system.

1. **Camera Tracking Services** track the position of marker objects based on the camera image sensor. The QoS attributes associated with these services are tracked error and response time. The corresponding Bayesian network for the service is shown in the Figure 4.3. We have used four Camera Tracking Services; two of them have pan and tilt capability (Wansview NCM625GA cameras), and the others are stationary webcams.

2. **Camera Adaptation Service** would tilt and pan the corresponding camera to get accurate tracking. The QoS attributes of the service are tilt error, pan error, and the response time. The Bayesian network for the service is shown in the Figure 6.8.

3. **Fusion Service** fuses the tracked positions from each camera using Kalman filtering [97] and provides an average tracked position of the marker.

4. **Fusion Adaptation Service** is a Replacement Adaptation Service (as discussed in section 6.2.2) that selects Camera Tracking Services that have the sight of the tracking object, and feed the output of the selected services along with their QoS to the Fusion Service.

The Adaptive Tracking System needs to modify its behavior with the movement of the marker, so that it can continuously deliver the required QoS and Trust values. We apply the proposed AdaptTrust model to this system in following ways:

First, we followed the four stages of the QoS and Trust prediction model (presented in Chapter 4) at the design time of the system. This process is equivalent to the case study mentioned in the section 5.1. After this step, each service has a trained Bayesian network, and the composed tracking system has a derived Bayesian network. Next, at the runtime of the system, we run the context adaptation algorithm described in Section 6.2.3.

6.2.4.1 Experiment 1: Tracking single marker

We have tracked a moving object in the (x, z) plane using the Adaptive Tracking System and recorded the actual versus average tracked positions of the object. This motion is plotted in Figure 6.12. Here, the difference between the actual position and the tracked position indicates the tracking error along the ‘x’ axis (for vertical lines) or ‘z’ axis (for horizontal lines). Additionally, the figure indicates the points of the context and replacement triggers, while the object is moving.

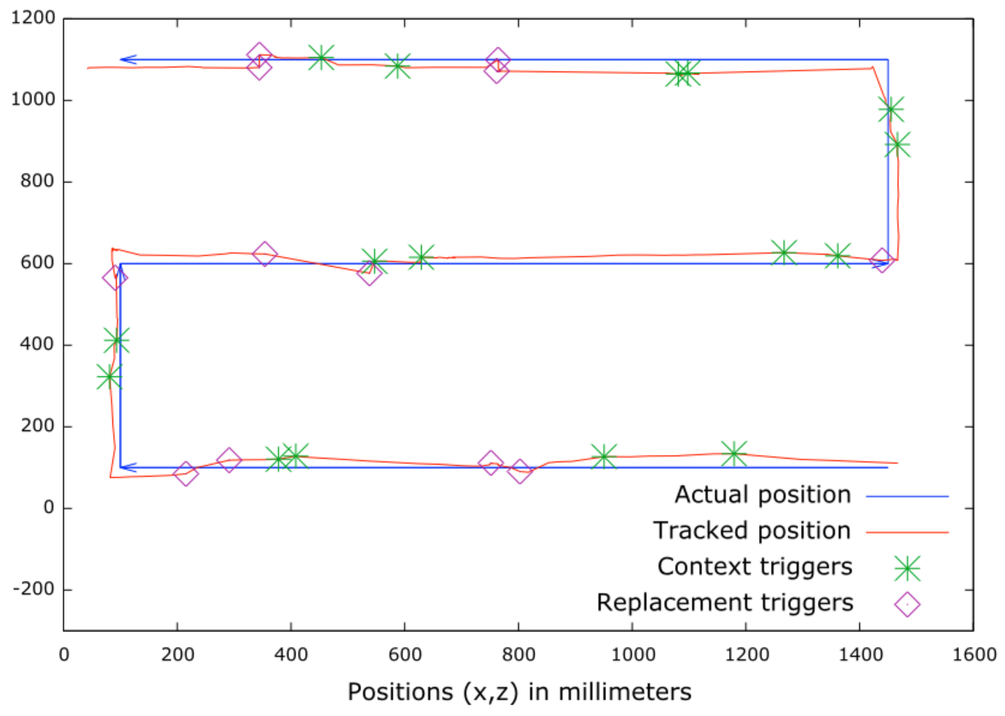


Figure 6.12.: Actual vs tracked positions and points of triggers.

From Figure 6.12, it can be seen that the triggers and the corresponding adaptations have a positive effect of the tracked position and they keep the tracking error around the threshold of the user requirements. Here, the points of context triggers represent positions where the predicted error (from the QoS and Trust prediction model) violates of the threshold and the adaptable cameras adjust its pan and tilt;

therefore, such adaptations have mostly resulted in better QoS values. Whereas the replacement triggers occur when the objects go out of the sight or appear back in the sight of the cameras; therefore, replacement adaptations have resulted in either better QoS (if the object appears in a camera) or worse QoS (if the object goes out of the sight of camera) values.

The average response times of the Adaptive Tracking System for different trigger situations at the steady state is shown in the Table 6.5. Note that the context trigger measurements exclude the time taken to physically pan and tilt the cameras, which takes 2-4 seconds, to focus on measuring the overhead associated with the AdaptTrust model.

Table 6.5.: Average response times of the system at triggers

No Triggers	**Replacement Triggers**	**Context Triggers (without heuristics)**	**Context Triggers (with heuristics)**
35.76ms	35.96ms	65.58ms	36.36ms

In Table 6.5, the response time of the Adaptive Tracking System with no triggers indicates the performance of the Tracking System and the adaptation algorithm up to step 2. Compared to that, the replacement triggers do not cause any overhead, as the response time of the Kalman fusion does not significantly change with the change of number of camera readings. In contrast, the context triggers cause the need to execute the backward inferencing algorithm and if no heuristics are used, it significantly increases the overall response time. However, with the use of sample caching heuristics, our model was able to minimize the overhead of the adaptation.

6.2.4.2 Experiment 2: Tracking two markers

In the second experiment, we have used the Adaptive Tracking System to track two markers moving at opposite directions. The Camera Adaptation Services are designed in a way that each camera follows the movement of one marker. Their movements and the tracked positions are shown in Figure 6.13.

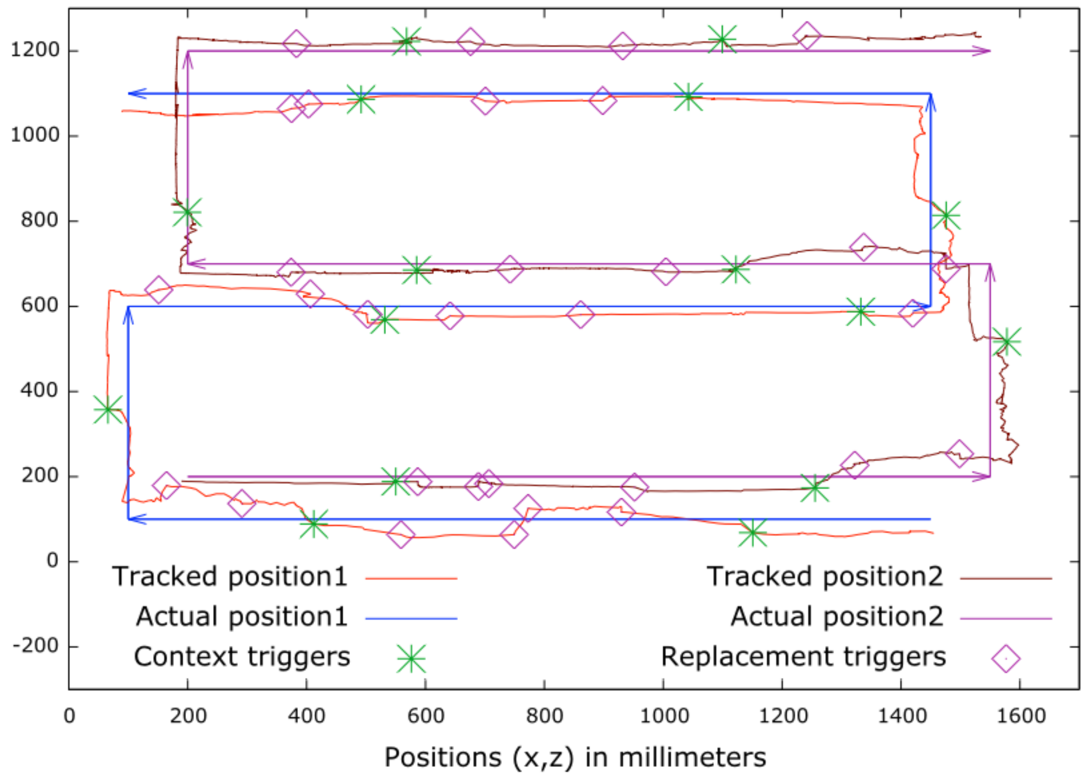


Figure 6.13.: Actual vs tracked positions for two markers.

Compared to the first experiment (Figure 6.12), in the second experiment (Figure 6.13), more replacement triggers occur. This is because only one adaptable camera is following a marker, and the other adaptable camera can track the same marker only when both markers are close to each other. Therefore, additional replacement triggers occur when two markers approach each other and separate from each other. Additionally, The figure 6.13 shows higher error values in the movements close to the left and right ends, as the markers are tracked by a lesser number of cameras at sides compared to the experiment 1. The response time of the Adaptive Tracking System, while tracking two objects is only slightly higher than tracking a single object. When there are no triggers, the response time is 35.79ms. This is because, the processing of the two markers take place in two parallel services, and the context, replacement triggers happened independently with each other, therefore the overhead of having two markers in our Adaptive Tracking System is negligible. These experiments show that the Adaptive Tracking System is continuously able to provide the QoS and Trust values within the user requirement thresholds using the proposed AdaptTrust model with a less overhead.

6.3 QoS and Trust Evaluation Model for Reused Services – ReuseTrust

The ContextTrust model proposed in Chapter 4 requires the system developers to identify the context-QoS dependencies of each participating services quantitatively at the initial phase. For that, developers need to use trace logs of the execution of the services. In practice, there are situations that developers do not have access to context-QoS dependency information about all individual services. However, they may have access to the trace logs of composed systems that use these services as part of the composition. Since the same set of services can be reused to develop multiple composed systems, we will be able to use the QoS and trust knowledge of these composed systems to derive the QoS and trust information about the individual services. We develop 'ReuseTrust' model to evaluate Bayesian networks of individual services from the Bayesian networks of the existing composed systems. These evaluations can also be used to identify bottleneck services that causes degradation of QoS and trust of existing systems, and replace them with services that improve QoS and trust of those systems. In this model, we assume that we know the Bayesian networks of some of the participating services. As the results below show, when all the participating services are unknown, the ReuseTrust model does not performs accurately.

6.3.1 ReuseTrust Algorithm

Let there be 'n' number of composed systems (C_1, C_2, $\ldots$, C_n) with the known context-QoS dependency information. These systems are composed of 'm' number of services (S_1^0, S_2^0, ..., S_m^0), which are referred as unknown services, and 'l' number of services (S_1^1, S_2^1, ..., S_l^1), which are referred as known services. We assume that we know the structures of the Bayesian networks of all the services and composed systems. Only the quantitative information about the dependencies is missing for the unknown services.

We refer all the dependent vertices of the unknown services (V_1^0, V_2^0, ..., V_p^0) as 'unknown vertices', and all the dependent vertices of the known services as 'known vertices' (V_1^1, V_2^1, ..., V_q^1). The rest of the vertices, which are all the independent vertices of the services (U_0, U_1, ..., U_r) are referred as context vertices. This set of systems is visualized in Figure 6.14.

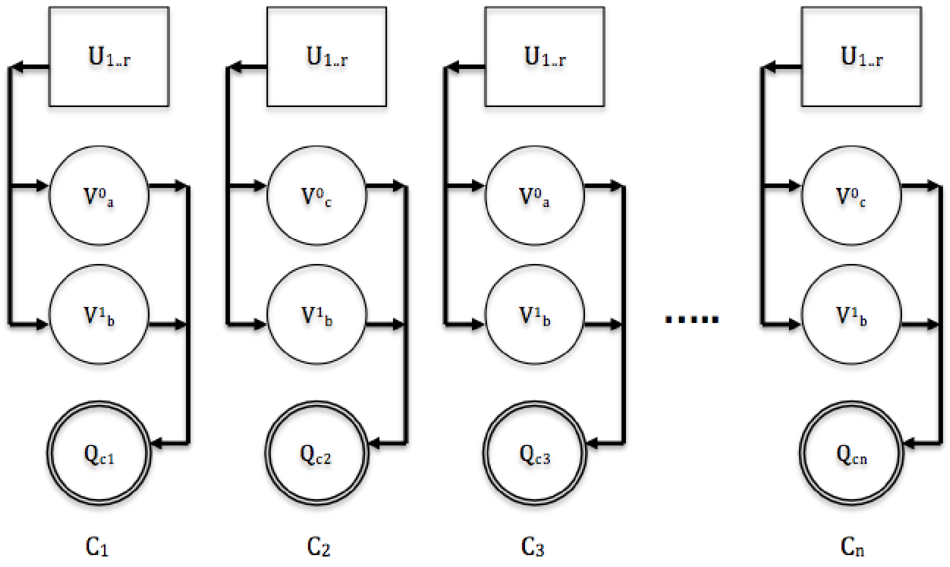


Figure 6.14.: An application of 'ReuseTrust' algorithm

We use the context values that composed systems are executed as inputs to the algorithm. We arrange the input data as a map from the context values to the mean and variance of QoS of each composed system. With this arrangement, an input instance looks like the expression 6.16:

$$(Ct_1, Ct_2, \dots, Ct_r) \rightarrow (M_{C_1}, VAR_{C_1}, M_{C_2}, VAR_{C_2}, \dots, \dots, M_{C_n}, V_{C_n}) \tag{6.16}$$

Here the left side of the arrow contains the values for 'r' context vertices. The right side of the arrow contains the mean (M_{C_i}), and variance (VAR_{C_i}) of the QoS for each of the n composed system. For simplicity, we assume each composed system has one QoS property. However, the algorithm can be generalized for multiple QoS properties by re-iterating the algorithm for each QoS attribute.

We assigned the context values of an input (i.e, Ct_i) to the corresponding context vertex (U_i). The algorithm described below should be run for each input instance iteratively.

We follow the Metropolis algorithm [58] to generate the required samples as described in following steps to infer the quantitative information about the dependencies of unknown vertices.

1. Initialize the mean (M_{V_i}) and variance (VAR_{V_i}) values for the proposal Gaussian distribution of each unknown vertex (V_i^0). Although, this can be initialized with random values, it will make the convergence slow. Therefore, for the mean(M_{V_i}), we assume each service contributes equally to the composed systems and divides the mean of the QoS of the composed systems equally among the participating services. For the variance (VAR_{V_i}), we use the variance of composed systems as upper bounds for the variance of the services.

2. Initialize the accepted samples as an empty set.

3. For iteration n, generate a sample for each unknown vertex (V_i^0) using the proposal Gaussian distribution with mean M_{V_i}, and variance VAR_{V_i}.

4. Infer each known vertex (V_i^1) from the context vertices (U_i) using the Bayesian networks of the individual known services (S^1) (i.e., forward sample within individual Bayesian networks).

5. Use Bayesian network of each composition (C_j), and the samples from the unknown vertices (V_i^0) and known vertices (V_i^1) to derive the samples for QoS of each composition. (i.e., forward sample in composed Bayesian network). There will a sample value V_{C_j} for the QoS of each composition C_j.

6. Calculate the expression E_n using Equation 6.17 from the QoS samples from each composition.

$$E_n = \sum_i \ln P_{N(M_{C_j}, VAR_{C_j})}(V_{C_j}) \tag{6.17}$$

   Note that we keep the probability values as logs to avoid the precision losses with very low probability values.

7. For iteration $n = 0$, we accept the sample without any condition. For iterations $n > 0$, we use the acceptance ratio (α) to decide whether to accept/ reject the sample.

$$\alpha = min(1, e^{E_n - E_{n-1}}) \tag{6.18}$$

   Note that since the variance (VAR_{V_i}) associated with an unknown vertex does not change with iterations, the proposal distribution is a symmetric probability distribution. Therefore, we can use the above acceptance ratio (α) to reject/accept the samples following the Metropolis algorithm.

8. Generate a random value (r) from uniform distribution $U(0, 1)$, and if $r < \alpha$, we accept the sample. Otherwise, a duplicate of the last ($r - 1$) set of samples is used as the set of samples of the current iteration (r). Then for unknown vertices, we assign the values V_i^0 to the M_{V_i} and repeat the steps 3-8 with iteration $n + 1$.

Here, we repeat the above steps for N number of iterations to obtain an adequate set of samples for each unknown vertices. We ignore the first M number of samples as the first set of samples may not converge to the expected distribution. Additionally, we skip R number of iterations to collect a valid samples to avoid some duplicate samples. The use of these parameter values generates $\frac{N-M}{R}$ number of samples. These parameters are tuned to get more efficient, and accurate predictions.

6.3.2 Results of the ReuseTrust Algorithm

To empirically validate the ReuseTrust algorithm, we use the Travel Planning System introduced in Section 5.2. The system contains 25 individual services as shown in Table 3.6. Each service has four endpoints with different associated security policies, similar to the case study used in Section 6.1.5. We use the tunning parameters shown in Table 6.6 to run the following experiments. With these parameters, simulation generates 1×10^5 number of samples that we can used to infer the QoS distributions of unknown services.

Table 6.6.: Tuning parameters of ReuseTrust algorithm

Parameters	**Value**
N	1.1×10^6
M	1×10^5
R	10

In the first experiment, we vary the number of unknown services from 1 to 25 and use 100 systems that are composed of these services. The selected set of compositions for the training set covers all the 25 services (i.e., Each of the 25 service is included in at least one composition of the training set). We apply the ReuseTrust algorithm to the training sets to evaluate the distributions of response times of unknown services under different context. Then, we calculate the mean absolute error of the means

(μ_{rt}) using the actual and predicted distributions of the response time of unknown services.

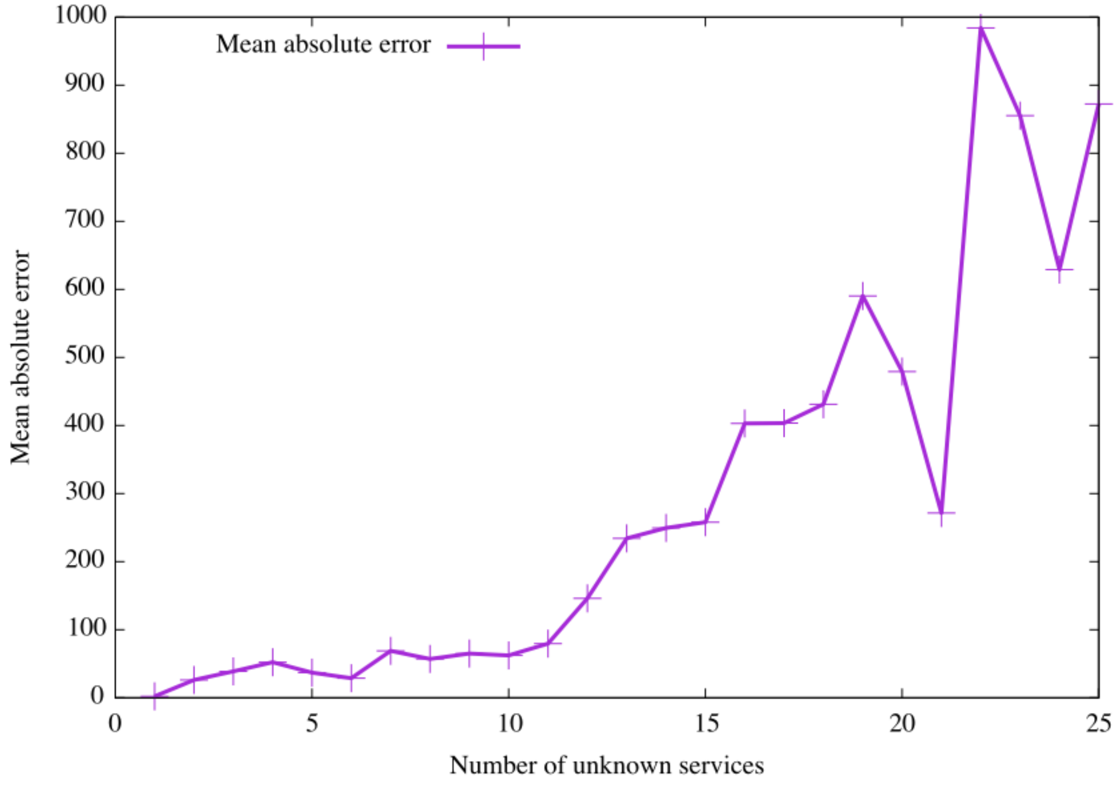


Figure 6.15.: The mean absolute error of the μ_{rt} with the number of unknown services

The Figure 6.15 shows that even up to 11 unknown services, the mean absolute error of the μ_{rt} stays less than 100. The error seems largely increase when the number of unknown services go up from 12 to 25. This shows that the algorithm is capable of predicting QoS of unknown services, in compositions, when around nearly half of the participating services have unknown QoS. The algorithm provides less accurate predictions when more unknown services are available. The reason for that is the algorithm can have multiple convergence values. Therefore, when performing random sampling the convergence can result in a different value than the actual value. As the number of known services increase, possible convergence values become close to the expected value as shown by the above experimental results.

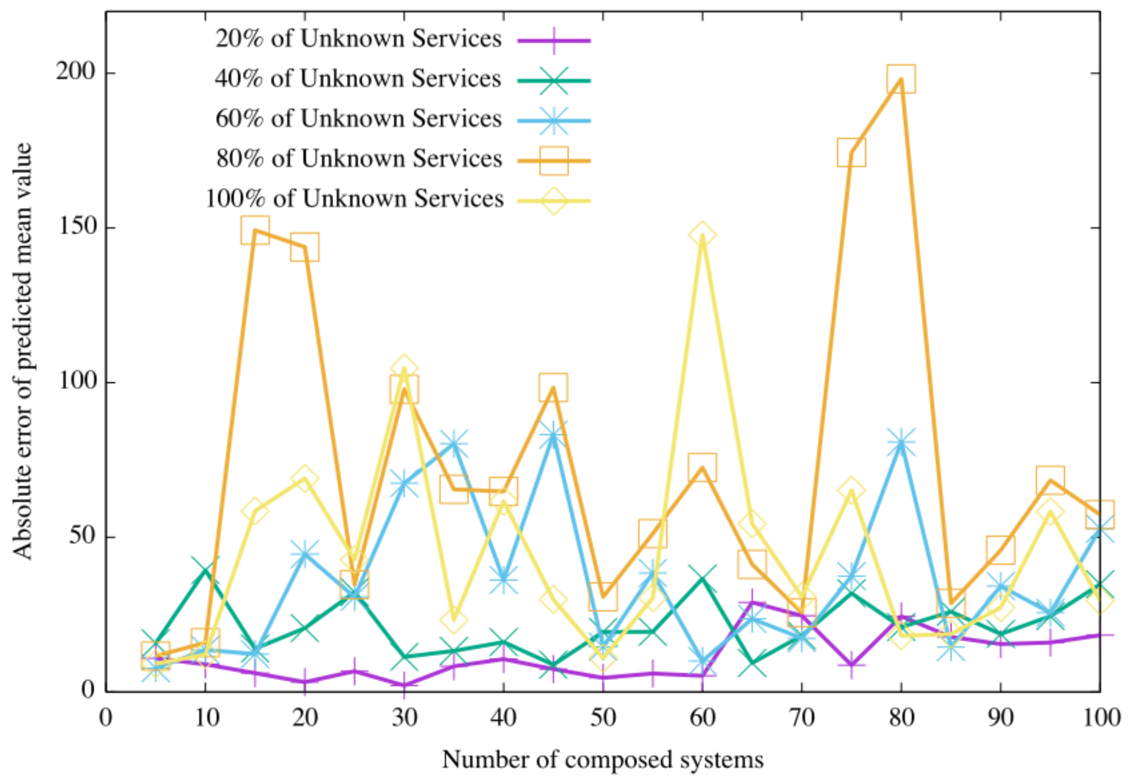


Figure 6.16.: The mean absolute error of the μ_{rt} with varying number of compositions

In the second experiment, we focus on the impact of the number of training sets to the accuracy of the predictions. There, we vary the training sets from 5 to 100, and mark different percentages (from 20% to 100%) of the participating services are unknowns. Then, we evaluate the mean absolute error of the μ_{rt} similar to the experiment 1. The results of the experiment are shown in Figure 6.16. It is clear that when the number of unknown services are at most 40% of total participating services, the proposed algorithm have given consistently good results. When the number of unknown services are more than half of the total participating services, there is more possibility the algorithm converges to a local optimum point, providing higher error values.

In this chapter, we have discussed another three applications of the ContexTrust model in addition to the predictions of QoS and trust that are discussed in earlier chapters. We have developed an extension to the ContexTrust model to select the optimum subset of services for a composed system using OptimumTrust model, build self-adaptive trustworthy system using AdaptTrust model, and evaluate the context-QoS dependencies of individual services that has been reused in existing composed systems using ReuseTrust model. We have also validated each of these models using case studies and demonstrated their effectiveness in real-life applications.

7 CONCLUSION

In this dissertation, we have developed basic models to predict QoS and trust of composed systems at the early phases of the system development lifecycle, and extended these basic models that suit requirements of different practical applications with QoS and trust concerns. Additionally, we have empirically validated the effectiveness of these models compared to prevalent models using real life case studies. The proposed models are developed under two frameworks: 1. Context independent QoS and trust prediction framework, 2. Context dependent QoS and trust prediction framework. The later framework is being developed from bottom up to address the drawbacks of the former framework as discussed in Section 3.3. The summary of the proposed models under each framework is listed below.

1. Context independent QoS and trust framework

   (a) BDUTrust model – To predict trust (as (B,D,U) measurements) of composed systems at the design phase of the system without considering the context.

   (b) RegressionTrust model – To predict QoS and Trust (B, D, U) values of future composed systems using existing composed systems that reuse the same set of participating services.

2. Context dependent QoS and trust framework

   (a) ContextTrust model – To predict QoS and trust values of composed system associated with certain context at the design phase of the system.

   (b) OptimumTrust model – To infer the subset of services that provide the optimum QoS and trust values of a composed system.

(c) AdaptTrust model – To adapt composed systems with the changes in context to maintain the required QoS and trust of the system.

(d) ReuseTrust model – To infer context to QoS and trust dependencies of individual services using existing systems that reuse the services.

We discuss the conclusions of each of the models in following sections.

7.1 BDUTrust Model

The BDUTrust model uses arithmetic and subjective logic operators chosen by domain experts to predict QoS and trust of composed systems at the design phase of the system development life-cycle. The main advantage of the model is that it does not require any training data to perform predictions. We use indoor tracking system case study to empirically validate the model. The results show that the predictions performed by the model have high uncertainty for individual services and composed systems, and as the complexity (number of services and interactions) of the composed systems grows, the uncertainty grows significantly. The main reasons for this issue are: no consideration of the context in the trust predictions, and use of subjective logic based trust representation, which does not provide a good representation of trust related to non-binary QoS properties (such as response time, and tracking error). We have discussed the lessons learned from the BDUTrust model in Section 3.3. Our solution is to build a new model named ContextTrust that considers the context in trust evaluations, and uses probability distributions to represent QoS and trust of non-binary QoS properties. We published our work on BDUTrust model at IEEE 15th International Conference on Computational Science and Engineering (CSE 2012) [17].

7.2 RegressionTrust Model

The RegressionTrust model uses linear regression techniques to predict the QoS and trust of composed systems. The model is trained using existing systems that reuse the same set of individual services as the future systems. The model is empirically validated using the travel planning system case study. The main assumptions of the model are: 1. The feature matrix of the model is left-invertible, which requires training data from many existing systems with different number of participating services. 2. The composition operator of the QoS properties is limited addition. The advantages of the model are: 1. The QoS properties and the associated trust values (B, D, U) of composition systems can be predicted with a high degree of accuracy using the evidences from other related compositions. 2. The model also provides information about how the association between two services impact on the system properties and the corresponding trust values. The model uses the knowledge of the association context to improve the performance of the predictions. However, it has the same drawbacks as BDUTrust model as it does not consider the other forms of context information, and uses the subjective logic based trust representation over probabilistic trust representation. We propose an improved model (named ReuseTrust) for similar applications with limited assumptions and addressing the drawbacks of the RegressionTrust model. We published our work on RegressionTrust model at International Conference on Network Infrastructure Management Systems (Interface 2013) [18].

7.3 ContexTrust Model

The ContexTrust model predicts the QoS and the trust values 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. The main advantage of the model is that we do not need training data from the composed system to obtain the corresponding composed Bayesian network. Therefore, we can simulate the QoS and trust behavior of the composed system before it is being built and make design decision in early phases of the system development life-cycle. We have presented the effectiveness of the proposed framework by model validations using three case studies: 1. indoor tracking system, 2. travel planning system, and 3. Collaborative bullying classification system. We were able to predict the QoS and trust values of the systems in case studies more accurately than the prevalent methods using the information available at the design time of the system. We published our work on ContextTrust model at IEEE International Conference on Web Services (ICWS 2015) [19].

7.4 OptimumTrust Model

The OptimumTrust model tackles the optimum service selection problem based on optimization algorithms and the ContextTrust model. The model selects the set of services to compose a system while optimizing the composed QoS and Trust of the system for an intended context. We propose heuristics to improve two prevalent optimization algorithms (Cross Entropy and Ant Colony Optimization) to consider the associations between services when solving the optimum service selection problem. We have validated the performance and the scalability of the proposed heuristics in solving the optimum service selection problem using simulation studies. Additionally, we use a case study, a travel planning system, to show the importance of consideration of context in solving the optimum service selection problem. We published our work on OptimumTrust model at International Journal of Services Computing (IJSC 2016) [20].

7.5 AdaptTrust Model

AdaptTrust model enhances the capabilities of the ContexTrust model to carry out inferences in real-time that will help the developers to make distributed systems trustworthy and self-adaptive. The model uses sensor services to detect the context changes, monitors whether the QoS and trust values deviate from the user specified levels, evaluates the necessary reactions using the context adaptation algorithm, and triggers adaptation services with necessary parameters to recover the system to satisfy the user QoS and trust requirements. The context adaptation algorithm uses heuristic methods to perform fast inferencing about adaptations triggers to be effective in situations where real-time adaptation is required. The process that continuously monitors the context uses the Kalman filter-based techniques to estimate the changing context with a higher accuracy over time. We have used an indoor tracking system as a case study to show the effectiveness of the model. We published our work on AdaptTrust model at IEEE International Conference on Service Computing (SCC 2016) [21].

7.6 ReuseTrust Model

The ReuseTrust model uses data from existing systems to infer QoS and trust properties of participating services. This model help identifying the context-QoS relationships of individual services, when the execution traces of the individual services are not available. We propose a sampling based algorithm that uses existing compositions as training set to derive the QoS distributions of unknown services under different contexts. The proposed model can also be used to automate the development of Bayesian networks corresponding to individual services that has been reused widely in existing systems. We use case studies to validate the algorithm empirically, and show that it provides accurate predictions of QoS of services, even when there are around 60% of the services used in the composed systems are unknown.

7.7 Lessons Learned

Prevalent approaches have proposed models that predict QoS of composed systems based on QoS of participating services and their interaction patterns. We have learned from our work, that it is also important to consider context information (along with the information used in prevalent approaches) to build more accurate prediction models for QoS and trust of composed systems. The proposed models require us to identify the context-QoS dependencies of participating services that make the composed system. This requires additional cost, time and effort up-front at the design phase of the system life-cycle. However, the model provides more accurate predictions about the QoS and trust behavior of the future systems than the prevalent alternatives. Such accurate predictions help the developers to make important design decisions at the early phase of the system life-cycle. As fixing issues at later phases, requires higher cost, time and effort that grows exponentially with time. We also learned that the context-QoS dependency information can be used to track the QoS and trust changes at the runtime and trigger reactions to keep the QoS and trust values at satisfactory levels.

7.8 Future Work

- **Validate the models with more case studies**: We have empirically validated the proposed models using three different application domains (i.e, indoor tracking, travel planning, and cyber-bullying detection). However, it is important to validate the models with other case studies from more domains in the future. Some example domains that can be used to validate the models are: the V2V communication domain, and the high performance computing domain.

- **Build models to work with partial knowledge of the context-QoS dependency information**: We have, in this dissertation, assumed that we have complete details about the context-QoS dependency information of the participating services and their interaction patterns up-front. It is important to study, in future, how well our models perform when no such information is available, or only partial information is available. Some of the information that we can use are subjective evidences such as user ratings and comments. Additionally, we can improve the accuracy of the predictions in real-time as new information is available using online machine learning techniques [102].

- **Study the dependencies from the context to the functional and synchronization behavior**: In this work, we have studied only the QoS behavior and trust associated with QoS values of the services and systems. In the future, the proposed models can be extended to study more aspects such as the functional and synchronization behaviors of the services and systems.

- **Improve the convenience of operating the models**: A drawback of the proposed models is that system developer requires more time at the design phase to collect evidences and capture the domain knowledge to perform necessary inferences. Therefore, there is a need to create collaborative and interactive tools to capture the domain knowledge of experts and developers. Additionally, such tools will be helpful for developers to generate code templates with the necessary QoS and trust-aware assertions that they can use as the basis for the composed system.

REFERENCES

REFERENCES

[1] Ashok Kumar Surekha Durvasula, Martin Guttmann. SOA practitioners' guide, part 1. http://www.soablueprint.com/whitepapers/SOAPGPart1.pdf, 2006.

[2] Andrew Martin. Trusted infrastructure 101. https://www.cylab.cmu.edu/tiw/slides/martin-tiw101.pdf, 2011.

[3] Audun Jøsang. Artificial reasoning with subjective logic. In *Proceedings of the 2nd Australian Workshop on Commonsense Reasoning*, volume 48. Perth, 1997.

[4] G. Shafer. *A Mathematical Theory of Evidence*, volume 76. Princeton university press Princeton, 1976.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454, January 2014.

[6] Valentin Cristea, Ciprian Dobre, and Florin Pop. Context-aware environments for the internet of things. *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence*, 460:25–49, 2013.

[7] Davy Preuveneers and Yolande Berbers. Internet of things: A context-awareness perspective. *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems*, pages 287–307, 2008.

[8] Gartner Inc. Gartner says the internet of things installed base will grow to 26 billion units by 2020. http://www.gartner.com/newsroom/id/2636073, 2013.

[9] ABI Research. More than 30 billion devices will wirelessly connect to the internet of everything in 2020. https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne, 2013.

[10] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the forty 7th Design Automation Conference*, pages 731–736. ACM, 2010.

[11] Sandra A Slaughter, Donald E Harter, and Mayuram S Krishnan. Evaluating the cost of software quality. *Communications of the ACM*, 41(8):67–73, 1998.

[12] Fabian Brosig, Nikolaus Huber, and Samuel Kounev. Automated extraction of architecture-level performance models of distributed component-based systems. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 183–192. IEEE, 2011.

[13] Chung-Wei Hang and Munindar P Singh. Trustworthy service selection and composition. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 6(1):5, 2011.

[14] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Transactions on the Web (TWEB)*, 1(1), May 2007.

[15] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1069–1075, New York, NY, USA, 2005. ACM.

[16] Yuan-sheng Luo, Yong Qi, Di Hou, Lin-feng Shen, Ying Chen, and Xiao Zhong. A novel heuristic algorithm for QoS-aware end-to-end service composition. *Computer Communications*, 34(9):1137–1144, 2011.

[17] Dimuthu U Gamage, Lahiru S. Gallege, James H. Hill, and Rajeev R. Raje. A compositional trust model for predicting the trust value of software system QoS properties. In *Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering*, pages 610–617. IEEE Computer Society, 2012.

[18] Dimuthu U Gamage, Lahiru S. Gallege, James H. Hill, and Rajeev R. Raje. Experimental evaluation of trustworthiness of compositional systems. *International Conference on Network Infrastructure Management Systems (Interface), Mumbai, India*, 2013.

[19] Dimuthu U Gamage, Lahiru S Gallege, and Rajeev R Raje. A QoS and trust prediction framework for context-aware composed distributed systems. In *2015 IEEE International Conference on Web Services (ICWS)*, pages 41–48. IEEE, 2015.

[20] Dimuthu U Gamage, Lahiru S Gallege, and Rajeev R Raje. Composing context-aware distributed systems using QoS and trust principles. *International Journal of Services Computing (IJSC)*, 4(2):32–48, 2016.

[21] Dimuthu U Gamage, Lahiru S Gallage, and Rajeev R Raje. A QoS and trust adaptation framework for composed distributed systems. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 251–258. IEEE, 2016.

[22] Michael C Jaeger, Gregor Rojec-Goldmann, and Gero Muhl. QoS aggregation for web service composition using workflow patterns. In *Enterprise distributed object computing conference, 2004. EDOC 2004. Proceedings. 18th IEEE International*, pages 149–159. IEEE, 2004.

[23] San-Yih Hwang, Haojun Wang, Jian Tang, and Jaideep Srivastava. A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. *Information Sciences*, 177:5484–5503, December 2007.

[24] Hisain Elshaafi, Jimmy McGibney, and Dmitri Botvich. Trustworthiness monitoring and prediction of composite services. In *2012 IEEE Symposium on Computers and Communications (ISCC)*, pages 580–587. IEEE, 2012.

[25] Vasu Alagar and Mubarak Mohammad. A component model for trustworthy real-time reactive systems development. In *Formal Aspects of Component Software Scope and Topics (FACS), Sophia-Antipolis*, 2007.

[26] John Buford, Rakesh Kumar, and Greg Perkins. Composition trust bindings in pervasive computing service composition. In *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops (PERCOMW)*, Washington, DC, USA, 2006. IEEE Computer Society.

[27] W. Sherchan, S. Nepal, J. Hunklinger, and A. Bouguettaya. A trust ontology for semantic services. In *2010 IEEE International Conference on Services Computing (SCC)*, pages 313–320, july 2010.

[28] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceeding of the IEEE Symposium on Security and Privacy*, pages 79–93, 1994.

[29] Aris Zakinthinos. *On the Composition of Security Properties.* PhD thesis, University of Toronto, 1996.

[30] Shi-guang Ju Hai-yang Li Song Chen, Cong-hua Zhou. Analysis for the composition of information flow security properties on petri net. *2010 2nd international conference on Information Science and Engineering (ICISE)*, pages 1859–1863, Dec 2010.

[31] Michel Charpentier and K. Mani Chandy. Examples of program composition illustrating the use of universal properties. In *Jos Rolim, editor, Parallel and Distributed Processing, LNCS 1586*, pages 1215–1227. Springer-Verlag, 1999.

[32] Michel Charpentier and K. Mani Chandy. Reasoning about composition using property transformers and their conjugates. In *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics (IFIP-TCS2000), Volume 1872 of Lecture Notes in Computer Science*, pages 580–595. Springer-Verlag, 2000.

[33] M. Mehdi, N. Bouguila, and J. Bentahar. Trustworthy web service selection using probabilistic models. In *2012 IEEE 19th International Conference on Web Services (ICWS)*, pages 17–24, June 2012.

[34] Evgeni Eskenazi, Re Fioukov, and Dieter Hammer. Performance prediction for component compositions. In *Proceeding of the International Symposium on Component-Based Software Engineering*, pages 280–293. Springer, 2004.

[35] K. Krogmann, M. Kuperberg, and R. Reussner. Using genetic search for reverse engineering of parametric behavior models for performance prediction. *IEEE Transactions on Software Engineering*, 36(6):865–877, Nov 2010.

[36] Marin Silic, Goran Delac, and Sinisa Srbljic. Prediction of atomic web services reliability based on k-means clustering. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 70–80. ACM, 2013.

[37] Nebil Ben Mabrouk, Sandrine Beauche, Elena Kuznetsova, Nikolaos Georgantas, and Valérie Issarny. QoS-aware service composition in dynamic service oriented environments. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, pages 1–20, New York, NY, USA, 2009. Springer-Verlag New York, Inc.

[38] Hua Guo, Fei Tao, Lin Zhang, Suiyi Su, and Nan Si. Correlation-aware web services composition and QoS computation model in virtual enterprise. *The International Journal of Advanced Manufacturing Technology*, 51(5):817–827, 2010.

[39] Lina Barakat, Simon Miles, and Michael Luck. Efficient correlation-aware service selection. In *2012 IEEE nineteenth International Conference on Web Services (ICWS)*, pages 1–8. IEEE, 2012.

[40] Jimin Li, Junbao Li, Aiguo An, and Zhenpeng Liu. Two-way trust evaluation based on feedback. In *2010 International Conference on Logistics Systems and Intelligent Management*, volume 3, pages 1910–1914. IEEE, 2010.

[41] Noura Limam and Raouf Boutaba. Assessing software service quality and trustworthiness at selection time. *IEEE Transactions on Software Engineering*, 36(4):559–574, 2010.

[42] Xu Wu. A stable group-based trust management scheme for mobile p2p networks. *International Journal of Digital Content Technology and its Applications (JDCTA)*, 5(2):116–125, 2011.

[43] Zheng Yan and Christian Prehofer. Autonomic trust management for a component-based software system. *IEEE Transactions on Dependable and Secure Computing*, 8(6):810–823, 2011.

[44] Enrico Bini, Thi Huyen Châu Nguyen, Pascal Richard, and Sanjoy K Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Transactions on Computers*, 58(2):279–286, 2009.

[45] Mark Panahi, Weiran Nie, and Kwei-Jay Lin. The design of middleware support for real-time SOA. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, pages 117–124. IEEE, 2011.

[46] M García Valls and P Basanta Val. A real-time perspective of service composition: Key concepts and some contributions. *Journal of Systems Architecture*, 59(10):1414–1423, 2013.

[47] Molka Rekik, Khouloud Boukadi, Nour Assy, Walid Gaaloul, and Hanene Ben-Abdallah. A linear program for optimal configurable business processes deployment into cloud federation. In *2016 IEEE International Conference on Services (SCC), San Fransisco, USA, June 27 – July 2, 2016*. IEEE, 2016.

[48] Kahina Bessai, Samir Youcef, Ammar Oulamara, Claude Godart, and Selmin Nurcan. Scheduling strategies for business process applications in cloud environments. *International Journal of Grid and High Performance Computing (IJGHPC)*, 5(4):65–78, 2013.

[49] Elio Goettelmann, Walid Fdhila, and Claude Godart. Partitioning and cloud deployment of composite web services under security constraints. In *2013 IEEE International Conference on Cloud Engineering (IC2E)*, pages 193–200. IEEE, 2013.

[50] Philipp Hoenisch, Christoph Hochreiner, Dieter Schuller, Stefan Schulte, Jan Mendling, and Schahram Dustdar. Cost-efficient scheduling of elastic processes in hybrid clouds. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 17–24. IEEE, 2015.

[51] Stephen Kell. A survey of practical software adaptation techniques. *Journal of Universal Computer Science*, 14(13):2110–2157, 2008.

[52] Danny Weyns, M. Usman Iftikhar, Didac Gil de la Iglesia, and Tanvir Ahmad. A survey of formal methods in self-adaptive systems. In *Proceedings of the 5th International C* Conference on Computer Science and Software Engineering (C3S2E)*, pages 67–79, New York, NY, USA, 2012. ACM.

[53] Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Stefano Iannucci, Francesco Lo Presti, and Raffaela Mirandola. MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. *IEEE Transactions on Software Engineering*, 38(5):1138–1159, 2012.

[54] Radu Calinescu, Lars Grunske, Marta Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. Dynamic QoS management and optimization in service-based systems. *IEEE Transactions on Software Engineering*, 37(3):387–409, 2011.

[55] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS–aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[56] Norha M Villegas, Gabriel Tamura, Hausi A Müller, Laurence Duchien, and Rubby Casallas. DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*, pages 265–293. Springer, 2013.

[57] Shih-Chun Lin, Ian F. Akyildiz, Pu Wang, and Min Luo. QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach. In *2016 IEEE International Conference on Services (SCC), San Fransisco, USA, June 27 – July 2, 2016*. IEEE, 2016.

[58] Christopher M Bishop et al. *Pattern Recognition and Machine Learning*, volume 1. springer New York, 2006.

[59] Christian Von Der Weth and Klemens Böhm. A unifying framework for behavior-based trust models. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 444–461. Springer, 2006.

[60] Prof. Tesler. The maximum of n random variables. `http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-151-probability-and-statistics-in-engineering-spring-2005/lecture-notes/app11_max.pdf`, 2010.

[61] J. Laurie Snell Charles M. Grinstead. Introduction to probability, 2nd revision. `http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/`, 1997.

[62] OAuth.net. The OAuth 2.0 authorization framework. `http://oauth.net/`, 2012.

[63] Microsoft MSDN. Chapter 1: Authentication patterns. http://msdn.microsoft.com/en-us/library/ff647374.aspx, 2005.

[64] Ryan Rybarczyk, Rajeev Raje, and Mihran Tuceryan. e-DTS 2.0: A next-generation of a distributed tracking system. *Proceedings of the International Conference on "On Demand Computing" (ICODC)*, pages 140–149, 2010.

[65] Google Inc. The Google directions API. https://developers.google.com/maps/documentation/directions, 2013.

[66] MapQuest Inc. MapQuest directions API. http://developer.mapquest.com/web/products/dev-services/directions-ws, 2013.

[67] MapQuest Inc. MapQuest open directions API web service. http://developer.mapquest.com/web/products/open/directions-service, 2013.

[68] MapQuest Inc. MapQuest traffic API. http://developer.mapquest.com/web/products/dev-services/traffic-ws, 2013.

[69] Expedia Inc. Expedia affiliate network. http://www.expediaaffiliate.com/index.php, 2013.

[70] Google Inc. Place search. https://developers.google.com/places/documentation/search, 2013.

[71] Yahoo Inc. Yahoo! local web services. http://developer.yahoo.com/local/, 2013.

[72] US National Weather Service. National digital forecast database (NDFD). http://graphical.weather.gov/xml/, 2013.

[73] HAMWeather Inc. Aeris weather API. http://www.hamweather.com/products/aeris-api/, 2013.

[74] Weather2 Inc. 2 day forecast weather API. http://www.myweather2.com/developer/apis.aspx?uref=becda844-8299-4bf6-899b-d771a92b9dbf, 2013.

[75] Weather Channel Inc. Weather on your site – The weather channel. http://www.weather.com/services/xmloap.html, 2013.

[76] World Weather Online Inc. World weather online, I/O docs: API documentation. http://developer.worldweatheronline.com/io-docs, 2013.

[77] Yahoo Inc. Yahoo! Weather RSS feed. http://developer.yahoo.com/weather/, 2013.

[78] Hotwire Inc. Rental car shopping API. http://developer.hotwire.com/docs/read/Rental_Car_Shopping_API, 2013.

[79] Aboli Phadke, Ryan Rybarczyk, Rajeev Raje, and Mihran Tuceryan. Incorporating mobile devices in indoor tracking. *International Conference on Network Infrastructure Management Systems (Interface 2014), Mumbai, India*, 2014.

[80] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[81] Kantilal Varichand Mardia, John M Bibby, and John T Kent. *Multivariate analysis.* Probability and mathematical statistics. Academic Press, London, 1979.

[82] Dan Kalman. A singularly valuable decomposition: The SVD of a matrix. *College Math Journal*, 27:2–23, 1996.

[83] Brendan J Frey. *Graphical Models for Machine Learning and Digital Communication.* MIT press, 1998.

[84] Nir Friedman, Moises Goldszmidt, et al. Discretizing continuous attributes while learning Bayesian networks. In *Proceedings of 27th International Conference on Machine Learning (ICML)*, pages 157–165, 1996.

[85] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[86] J Scott Armstrong and Fred Collopy. Error measures for generalizing about forecasting methods: Empirical comparisons. *International journal of forecasting*, 8(1):69–80, 1992.

[87] Deepal Jayasinghe. Axis2 session management. `http://www.developer.com/services/article.php/3620661/Axis2-Session-Management.htm`, 2015.

[88] Apache Software Foundation. Apache Axis2/Java – Next generation web services. `http://axis.apache.org/axis2/java/core`, 2015.

[89] Apache Software Foundation. Apache Rampart - Axis2 security module. `http://axis.apache.org/axis2/java/rampart/`, 2015.

[90] Apache Software Foundation. Apache Neethi. `http://ws.apache.org/neethi/`, 2015.

[91] A. Mangaonkar, A. Hayrapetian, and R. Raje. Collaborative detection of cyberbullying behavior in Twitter data. In *2015 IEEE International Conference on Electro/Information Technology (EIT)*, pages 611–616, 2015.

[92] David L. Olson and Dursun Delen. *Advanced Data Mining Techniques.* Springer Publishing Company, Incorporated, 1st edition, 2008.

[93] Haipeng Guo and William Hsu. A survey of algorithms for real-time bayesian network inference. In *AAAI/KDD/UAI02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems.* Edmonton, Canada, 2002.

[94] Dirk P. Kroese. A tutorial on the Cross-Entropy method. *Annals of Operations Research*, 2005.

[95] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the 1st European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[96] Jianguo Ding. Probabilistic inferences in Bayesian networks. *arXiv preprint arXiv:1011.0935*, 2010.

[97] Wikipedia. Kalman filter – Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Kalman_filter`, 2015. [Online; accessed 28-February-2016].

[98] Wikipedia. Markov property – Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Markov_property`, 2015. [Online; accessed 28-February-2016].

[99] Wikipedia. Gibbs sampling – Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Gibbs_sampling`, 2015. [Online; accessed 28-February-2016].

[100] Ryan Rybarczyk, Rajeev R. Raje, and Mihran Tuceryan. eDOTS 2.0: A pervasive indoor tracking system. In *The 25th International Conference on Software Engineering and Knowledge Engineering, Boston, MA, USA, June 27-29, 2013.*, pages 429–434, 2013.

[101] Dimuthu Gamage. Adaptive indoor tracking system. `https://youtu.be/dCT1wNSAvlY`.

[102] Wikipedia. Online machine learning – Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Online_machine_learning`, 2015. [Online; accessed 02-December-2016].

VITA

VITA

Dimuthu Gamage received his Bachelor of Science degree from the University of Moratuwa, Sri Lanka in 2007 and Master of Science degree from the Purdue University, Indianpolis in 2014. Dimuthu has worked at Amazon Inc. and Google Inc. as an intern, and WSO2 Inc. as a software engineer. His current research interests are Software Engineering and Distributed Computing, more specifically on quality-aware and trust-based service composition and generative programming.