TRUST MANAGEMENT FOR CONTEXT-AWARE COMPOSITE SERVICES

AFAF MOUSA

A THESIS
IN
THE DEPARTMENT
OF
THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY (INFORMATION AND SYSTEMS ENGINEERING)
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

JANUARY 2021

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Afaf Mousa**

Entitled: **Trust Management for Context-Aware Composite Services**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Constantinos Constantinides

_____ External Examiner
Dr. Muhammad Younas

_____ Examiner
Dr. Juergen Rilling

_____ Examiner
Dr. Rachida Dssouli

_____ Examiner
Dr. Roch Glitho

_____ Supervisor
Dr. Jamal Bentahar

_____ Co-supervisor
Dr. Omar Alam

Approved by _____
Dr. Mohammad Mannan Graduate Program Director

January 18, 2021
Date of Defence

_____
Dr. Mourad Debbabi Dean, Faculty of Engineering and Computer Science

Abstract

Trust Management for Context-Aware Composite Services

Afaf Mousa, Ph.D.
Concordia University, 2021

In the areas of cloud computing, big data and internet of things, composite services are designed to effectively address complex levels of user requirements. A major challenge for composite services management is the dynamic and continuously changing run-time environments that could raise several exceptional situations such as service execution time that may have greatly increased or a service that may become unavailable. Composite services in this environmental context have difficulty securing an acceptable quality of service (QoS). The need for dynamic adaptations to be triggered becomes then urgent for service-based systems. These systems also require trust management to ensure service level agreement (SLA) compliance. To face this dynamism and volatility, context-aware composite services (i.e., run-time self-adaptable services) are designed to continue offering their functionalities without compromising their operational efficiency to boost the added value of the composition.

The literature on adaptation management for context-aware composite services mainly focuses on the closed world assumption that the boundary between the service and its run-time environment is known, which is impractical for dynamic services in the open world where environmental contexts are unexpected. Besides, the literature relies on centralized architectures that suffer from management overhead or distributed architectures that suffer from communication overhead to manage service adaptation. Moreover, the problem of encountering malicious constituent services at run-time still needs further investigation toward a more efficient solution. Such services take advantage of the environmental contexts for their benefit by providing unsatisfying QoS values or maliciously collaborate with other services. Furthermore, the literature overlooks the fact that composite services data is relational and relies on propositional data (i.e., flattened data containing the information without the structure). This contradicts with the fact that services are statistically dependent since QoS values of service are correlated with those of other services.

This thesis aims to address these gaps by capitalizing on different methods from software engineering, computational intelligence and machine learning. To support context-aware composite services in the open world, dynamic adaptation mechanisms are carried out at design-time to guide the running services. To this end, this thesis proposes an adaptation solution based on a feature model that captures the variability of the composite service and deliberates the inter-dependency relations among QoS constraints. We apply the master-slaves adaptation pattern to enable coordination of the self-adaptation process based on the MAPE loop (Monitor-Analysis-Plan-Execute) at run time. We model the adaptation process as a multi-objective optimization problem and solve it using a meta-heuristic search technique constrained by SLA and feature model constraints. This enables the master to resolve conflicting QoS goals of the service adaptation. In the slave side, we propose an adaptation solution that immediately substitutes failed constituent services with no need for complex and costly global adaptation. To support the decision making at different levels of adaptation, we first propose an online SLA violation prediction model that requires small amounts of end-to-end QoS data. We then extend the model to comprehensively consider service dependency that exists in the real business world at run time by leveraging the relational dependency network, thus enhancing the prediction accuracy. In addition, we propose a trust management model for services based on the dependency network. Particularly, we predict the probability of delivering a satisfactory QoS under changing environmental contexts by leveraging the cyclic dependency relations among QoS metrics and environmental context variables. Moreover, we develop a service reputation evaluation technique based on the power of mass collaboration where we explicitly detect collusion attacks. As another contribution of this thesis, we introduce for the newcomer services a trust bootstrapping mechanism resilient to the white-washing attack using the concept of social adoption. The thesis reports simulation results using real datasets showing the efficiency of the proposed solutions.

Acknowledgments

First and foremost, I offer my utmost gratitude to Ph.D. supervisors Dr. Jamal Bentahar and Dr. Omar Alam, for continuous support. Their immense knowledge and valuable insights have given me more power and spirit to excel in research writing. Our discussions have made every encounter an opportunity to improve the research quality.

I gratefully acknowledge the funding received towards my Ph.D. from the Egyptian government. I am also grateful for the funding received through Concordia University to undertake my Ph.D.

From the bottom of my heart, I would like to say big thank you to my family for their patience, inspiration, immeasurable and love. I promise to continue to warrant your faith in me.

I would like to thank everybody who was important to the successful realization of this thesis, as well as expressing my apology that I could not mention personally one by one.

Finally, I appreciate all the ups and downs, highs and lows, great days and horrible ones during the Ph.D. journey. I hold onto the hope I started with and power through each challenge. Thank you, Lord God Almighty, for the gift of faith. I know I am going to make it, as long as I don't stop.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the context of our research and presents the addressed problem. Then, it outlines the research questions that our work aims to answer. Finally, it identifies the objectives and contributions of the thesis.

1.1 Research Context and Motivations

Services in cloud, big-data, and IoT applications attracted the attention of the research community to deal with the continuous development and deployment of business processes. Modern applications rely on services as components that can interact with one another to deliver complex tasks through service composition. Service composition can be defined as a process that provides functionalities that were not available or defined at design time by compiling value-added services from individual services [29]. Therefore, a composite service can be considered as a complete software solution. The growing complexity of composite services to effectively address complex levels of user's requirements and the increasing emphasis on quality of service (QoS) necessitate solutions to deal with this complexity and ensure service level agreement (SLA) compliance in dynamic environments.

To face this dynamism and volatility, context-aware services are designed to continue offering their functionalities without compromising their operational efficiency capitalizing on autonomic computing. Therefore, it is important to support the trust-based dynamic adaptation for context-aware composite services to boost and secure the added value of the composition.

Although numerous trust models for composite services can be found in the literature, they overlook the contextual run-time environment of services that raises additional challenges to be tackled by trust models. Moreover, these models fail to comprehensively consider service dependency that exists in the real business world at run time. Collaboration among constituents in services composition creates different service dependency relations. The knowledge about dependencies among services ensures successful provisioning of the composite services. Furthermore, related work on the dynamic adaptation of composite services lacks support for analyzing the inherent variability of dynamic adaptation at design time to guide adaptation and face arising context changes in an unpredictable open world [2]. Thus, the motivation of this thesis is to design trustworthy context-aware composite services that can tackle unanticipated changes in the open world. Consequently, this thesis proposes an adaptation architecture considering these aforementioned challenges. As Figure 1 shows, we adopt the master-slaves adaptation pattern to enable coordination of the self-adaptation process for context-aware composite services controlled by the MAPE (Monitor-Analysis-Plan-Execute) loop.

Such an architecture enables context-aware composite services to effectively offer their functionalities in a dynamic open world. The hierarchical structure of the proposed architecture allows us to address the adaptation process of context-aware composite services at two levels: (1) process level to capture the whole picture of the business logic that will guide the high-level adaptation plan; and (2) service-level to achieve the ambitions outlined in the high-level adaptation plan. Our approach first replaces failed services at

the service-level adaptation, which could avoid unnecessary complete process reconfigurations. Accordingly, by considering both adaptation levels, we reduce the adaptation time complexity. Moreover, the proposed architecture enables us to maintain the service trust and reputation by monitoring and prediction QoS degradation for triggering appropriate adaptation actions to meet the SLA requirements.

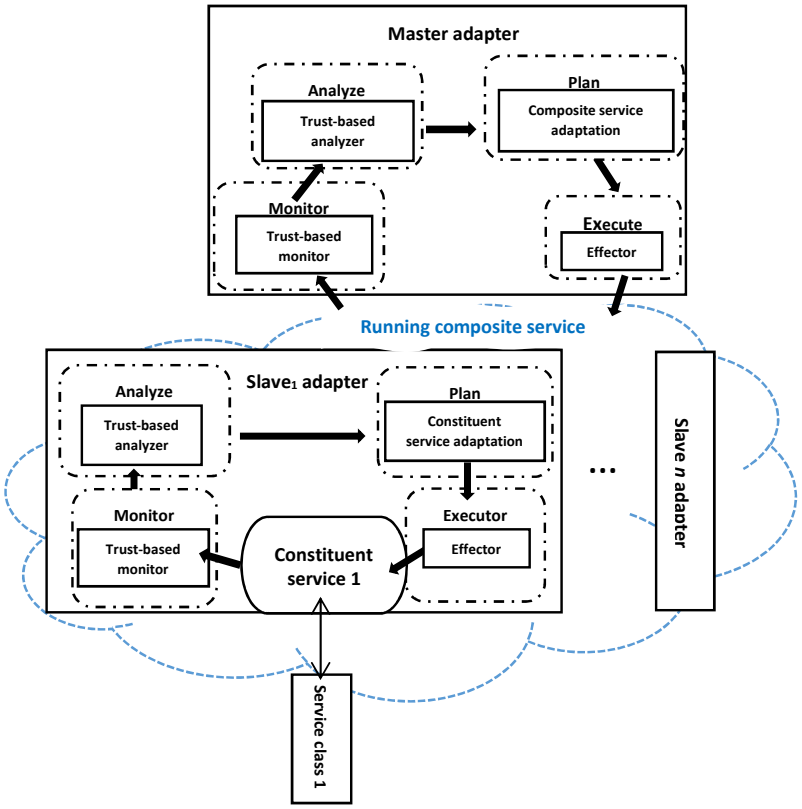


Figure 1: Our proposed adaptation architecture

1.2 Problem Statement and Research Questions

The dynamics of composite services in cloud and IoT settings highlights the importance of services with self-adaption capabilities that ensure and maintain user satisfaction. A review of the literature indicates that some problems still need to be considered. First, there is a

need for a dynamic adaptation approach with the minimal performance bottlenecks. Second, there is a need for a solution to handle the complexity of service composition reconfiguration with a search space of numerous possible configurations. Third, it is necessary to adapt against unknown changes in the open world in order to secure the SLA. Finally, besides the investigation of self-adaption capabilities of composite services to adapt to the context changes, it is necessary to investigate a sophisticated method to manage service trust and reputation and avoid malicious performance as regards providing unsatisfactory QoS values or using a contextual environment for the service advantage.

We formulate the main research questions this thesis aims to answer as follows:

- **RQ**1: How can we dynamically adapt service compositions to secure SLA in the open world?
- **RQ**2: How can we maximize the probability of detecting malicious performance at run-time to ensure the trustworthiness of the context-aware composite service?
- **RQ**3: How can we control SLA management of composite services at run-time in a relational setting to avoid high economic compensation caused by breach SLA?
- **RQ**4: How can we reconfigure a composite service to enhance its performance and resolve conflicting QoS goals while being limited by SLA and configuration constraints?

1.3 Research Aim, Objectives and Challenges

The goal of our research work is to provide a trust-based dynamic adaptation approach to restore and maintain QoS of context-aware composite services at run-time. To this end, we frame the following objectives:

- **Objective** 1: Develop a decentralized adaptation architecture that enables the distribution of the control of the adaptation process among different managers without communication and management overheads.

- **Objective** 2: Develop context-aware trust prediction models to detect malicious performance at run-time and support adaptation managers in their decisions making.

- **Objective** 3: Put forward an SLA management model that is able to learn service dependency in a relational setting.

- **Objective** 4: Introduce a context-aware adaptation approach that allows services in an open world to monitor, detect, predict, and decide about the accurate adaptation actions to meet the current encountered context changes while maintaining the user's satisfaction controlled by different types of constraints.

There are several challenges associated with these objectives:

- **Challenge 1: Adaptation management**. Although current research proposals have paved the way towards dynamic adaptation of service compositions, most solutions that tended to implement dynamic adaptations operate on centralized environments which lack (1) efficiency for a large-scale environment such as cloud, and (2) flexible and effective adaptation towards changes in the run-time environment [16, 17]. This is since a single controller is inadequate for adaptation and monitoring the overall performance. Furthermore, composite services are offered by distributed providers and reside beyond the domain of any single controller. Subsequently, the distributed nature of services plays against centralized adaptation controllers found in the literature. In an attempt to overcome the drawbacks of centralized approaches, some distributed approaches have been put forward [15, 60]. Unfortunately, these approaches result in communication overhead, which is detrimental to adaptation performance.

Therefore, there is a need for a dynamic adaptation approach with the minimal performance bottlenecks.

- **Challenge 2: Variability management**. A computing infrastructure that provides support for dynamic adaptation of service compositions is highly desirable and yet to be introduced. To this end, it is necessary to prepare the service mashup at design time. This will facilitate prompt responses for dynamic adaptation to secure critical systems based on service composition. Thus, it is required to capture the variability of composite services such that each possible configuration is verified at design time to avoid invalid configurations during execution. A composite service may have alternative variants that provide different QoS values. Therefore, in the advent of problematic changes, it can add a new constituent service or discard others in response to encountered changes in the running environment, hence, delivering a new service composition configuration. Due to the large number of possible configurations, service composition reconfiguration is a complex problem.

- **Challenge 3: Unpredictable open world**. Current research works have focused on the dynamic adaptation of service compositions in the closed world. In this closed setting, it is assumed that the boundary between a service and its context is known ahead of time and unchanging [8]. Thus, a set of adaptation actions is predefined for fully foreseen contexts [15, 16, 17, 60]. However, services run in an unpredictable open world. Accordingly, services should be able to react in face of continuous and unanticipated changes in uncertain contexts.

- **Challenge 4: Dynamic management**. There is a need to investigate the self-adaption capabilities of composite services to adapt to the contextual changes. In particular, a composite service must be able to (1) self-optimize service selection according to

the required SLA; (2) self-configure its components according to a possibly changing environment; (3) self-heal its components and the workflow it serves in case of failures; and (4) self-protect its provided QoS.

- **Challenge 5: Trust management**. At run-time, QoS of context-aware composite services is subject to changes or even failures. Since multiple constituent services are engaged in the composition, malicious services are expected to be involved. The environments dynamics could lead services to misbehave by unilaterally deviating from the agreements made upon service composition. Practically, services could provide different QoS values in different context environments. Moreover, constituent services may maliciously collaborate, which degrade the performance of the composite service. Furthermore, newborn services that may provide better QoS could not be involved in the composition due to the lack of resources to estimate their trust. Therefore, a sophisticated method to estimate service trust to avoid malicious performance is required.

- **Challenge 6: Composite service trust**. In a composite service, constituents' collaboration creates different service dependency relations which makes it challenging for SLA management at run-time. Knowledge about dependencies among services allows service providers to secure an acceptable SLA. Unfortunately, service dependency information is only implicitly described in the SLA. In addition, composite services data is relational, yet current approaches ignore the relations among individual services and rely on propositional data assuming that constituents are homogeneous and statistically independent. Representing real-world data as homogeneous, independent and identically distributed instances leads to statistical bias in the results. Therefore, it is necessary to allow an SLA management model to learn service dependency in a relational setting.

Figure 2 draws the overall overview of our research work (methodology, challenges,

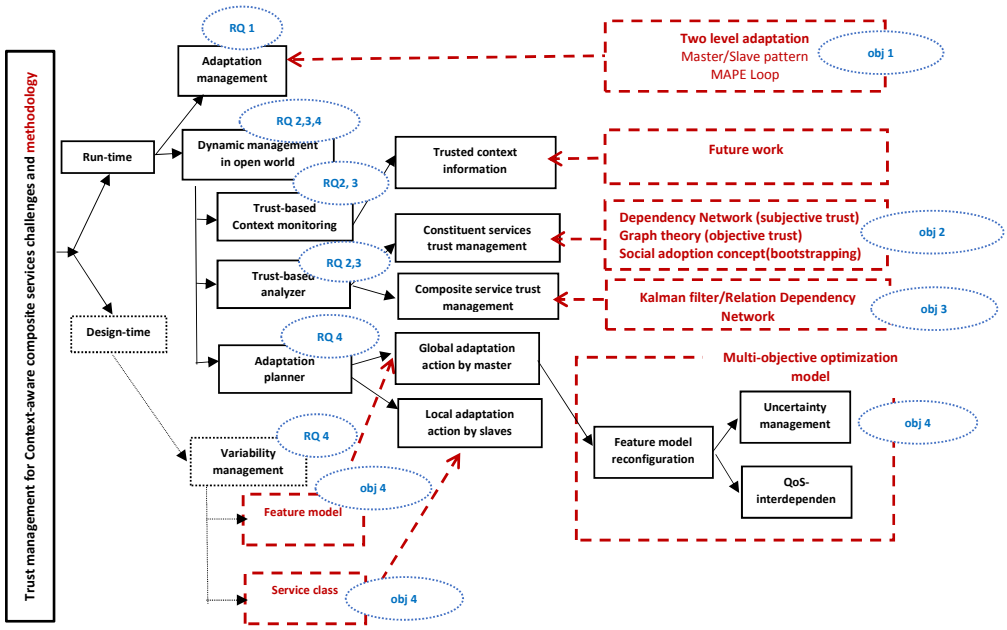


Figure 2: Research methodology, objectives and research questions

and objectives) w.r.t the identified research questions.

1.4 Research Contributions

The contributions of this thesis that accomplish the identified objectives are as follows:

- **Contribution 1**: We introduce a two-level adaptation process for composite services using the master/slaves pattern that enables the service provider to monitor QoS performance and trigger prompt adaptation actions. This contribution is discussed in Chapter 3.

- **Contribution 2**: We explore the cyclic dependency relations among QoS metrics and context variables and introduce a dependency network-based service trust model. This trust model enables slaves responsible for constituent services to estimate services trust considering cyclic dependency relations and the evolving nature of the

running environment. This contribution is examined in Chapter 4.

- **Contribution 3**: We develop an objective trust evaluation technique that enables slaves to detect collusion attacks and minimize the number of malicious services. In addition, we introduce for the newcomer services a trust bootstrapping mechanism that makes slaves more resilient to the white-washing attacks. This contribution is presented in Chapter 4.

- **Contribution 4**: We put forward an SLA management model that enables the master, which manages the composite service, to predict SLA violation and maintain service trust by securing the provided QoS. This contribution is exposed in Chapter 4.

- **Contribution 5**: At the master side, we model the adaptation process as a multi-objective optimization problem that resolves conflicting goals of minimizing the cost and maximizing the performance of composite services by considering the SLA and feature model constraints followed by a genetic-based algorithm that solves the optimization problem. At the slaves side, we introduce a local service adaptation action that promptly substitutes the failed constituent services to maintain the overall performance and reduce the need for global adaptation. This contribution is discussed in Chapter 5.

1.5 Research Assumptions

The following assumptions are made regarding this thesis:

- We assume service availability for service substitution by the slaves.

- We assume feature availability for service reconfiguration by the master.

- We assume services face unknown context changes in the run time environment, i.e. we have an open world run environment.

1.6 Thesis Structure

We provide in Chapter 2 the related background knowledge of this thesis. This chapter covers the foundation of trust-based autonomic service adaptation. Afterwards, we present the related state-of-the-art.

In Chapter 3, we present an overview of the proposed approach. We discuss the main tasks of the master and slaves toward the adaptation process through the MAPE loop.

In Chapter 4, we discuss the analyze step of the MAPE loop. We address the prediction models that enable the master and slaves to predict SLA/QoS violation for composite/constituent services, which support the decision making about the adaptation actions.

In Chapter 5, we discuss the plan step of the MAPE loop. We address the adaptation actions that are triggered to secure composite/constituent services by the master and slaves.

Finally, in Chapter 6, we summarize the thesis contributions and highlight its limitations and future work.

Chapter 2

Background and Literature Review

This chapter lays the essential foundations for this thesis. We present the following disciplines: (1) context-aware composite services, (2) autonomic computing, and (3) software product line engineering. Then, we present a literature review in the research areas that this thesis aims to address.

2.1 Context-aware Composite Service

2.1.1 Service Compositions

To describe the concept of service compositions, first, it is necessary to understand the main related concepts, namely those of services, quality of service (QoS), and service level agreement (SLA).

Services are self-contained applications that provide online services to facilitate loosely-coupled distributed business integration. To this end, service providers publish services by registering service information. Then, the users of the services discover the services to find the appropriate services that provide the required functionality. Upon discovery, the user requests the functionality by providing the required input. The service responds to the users

with the desired output.

Quality of service represents the set of those quantitative and qualitative characteristics of the service to achieve the required functionality [78]. Since numerous competitive services provide similar functionalities, QoS has become a decisive factor to distinguish the reputation of services. Thus, services are selected for compositions using QoS. Typical QoS metrics include response time, availability, throughput and probability of success [97]. Different QoS metrics describe service quality in different ways. For example, short response time (negative metric) is preferable for services while high availability (positive metric) is required.

QoS constraints (i.e, Service Level Objectives (SLOs)) are defined in service-level agreements (SLA) between services providers and users. SLA is a commitment that governs the association between a service provider and a user and indicates the expected level of service and the related expenses [13]. Besides, SLA defines monetary penalties in case of any violation of the written agreement.

2.1.2 Context

Context is information about the present environment [48, 61]. Specifically, context provides information that characterizes the current situation to provide the appropriate services. Context-awareness is the ability to extract, interpret and use context information and adapt the functionality to the current context [2]. In other words, a context-aware system acquires and utilizes information in the context to provide the appropriate services.

Accordingly, this thesis considers the context as the state of the running environment and the context variable as the environmental conditions that affects the behaviour of the service.

2.1.3 Service Dependency

Service composition is a collaborative process where constituent services cooperate to achieve a complex business goal. Dependencies exist between these collaborative constituent services. Knowing about these dependencies allows composite service providers to manage SLA to ensure successful provisioning of the composite service.

Service dependency can be classified as control flow dependency that exists from the business flow, semantic dependency that occurs through the domain ontology, message dependency that happens by sending or receiving input and output messages among services, and QoS dependency which this thesis focuses on. In particular, we consider three different kinds of QoS dependencies: horizontal, vertical and cyclic dependencies. Horizontal dependencies occur between constituent services, e.g., violations or changes to the QoS values of one service affect QoS of the dependent services. Vertical dependencies exist between the composite service and its constituents, i.e., QoS values of the composite service are affected by the QoS values of constituents. This thesis explores cyclic dependencies among the QoS values of individual and composite services in a dependency chain (loop), e.g., response time depends on price and vice versa.

2.2 Autonomic Computing

In consideration of the need for flexible, resilient, dependable, recoverable, customizable, configurable, and self-optimizing software systems, self-adaptation has become an important research topic. IBM in 2001 proposed Autonomic computing (AC) to develop systems with self-management capabilities [45]. Self-Adaptive Software (SAS) evaluates and changes its behaviour to correctly achieves its goal or enhances its performance [30]. IBM defined four self-properties of AC: self-healing, self-configuring, self-optimizing and self-protecting. These properties come from biological self-adaptation mechanisms, e.g. the

human body adapts itself to changes in its context (changing environment temperature) [1]. These properties are described as follows:

- Self-healing is the ability of discovering, diagnosing and reacting to disruptions. A service predicts the potential problems and repairs itself to prevent its failure which ensures its availability, survivability, maintainability and reliability.

- Self-optimizing is the ability to monitor and manage resource allocation to meet user's requirements. Services should have the ability to improve their QoS values to fulfill the SLA constraints, i.e. efficiency and functionality.

- Self-configuring is the ability of dynamically adapting to context changes by reconfiguring software entities according to high-level goals.

- Self-protecting is the ability to detect, identify and protect against malicious behaviours. This makes the services less vulnerable and ensures their reliability and functionality.

2.3 MAPE Loop

For controlling self-adaptable systems, IBM proposed an autonomic control loop, i.e., MAPE loop [45], that is an architectural framework of four steps. Figure 3 illustrates the MAPE control loop steps: monitor, analyze, plan, and execute steps. The loop is completed by connecting to the adaptable system through sensors and executers. In the context of this thesis, an adaptable managed system is the context-aware composite service, while the autonomic manager is a software layer supervising the service. The manager goes through MAPE steps:

- Monitor: This step collects data and monitors the running service through sensors. Those sensors could be external services to detect the existence of context changes

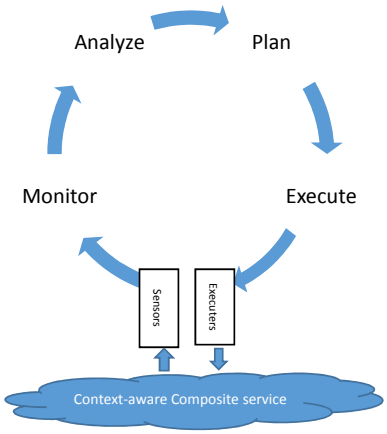


Figure 3: MAPE control loop

that could include:

  - Addition/removal of a task from the composite service.
  - The unavailability of constituent service, as it can leave the run-time environment dynamically.
  - Discovery of new constituents with better QoS.
  - QoS degradation

- Analyze: This step converts the collected data by the monitor step to behavioural patterns and symptoms. It performs statistical computation to predict violation in some QoS constraints defined in SLA, e.g. the measured response time exceeds its SLA constraint.

- Plan: This step decides the adaptation actions to be taken to achieve the best performance based on the data provided by the previous step. The adaptation action could reconfigure the workflow of the composite service, or it could select new constituent services to implement the needed functionalities.

- Execute: This step applies the adaptation actions on managed services. The adaptation actions are executed by binding and unbinding services.

2.4 Software Product Line and Variability Modeling

Software product lines (SPL) is a family of software systems that satisfy the specific needs of a particular market segment or mission with some commonalities and significant variabilities [32]. A commonality is quality or functionality that is share. In contrast, variability is the capability to change or customize a system through variation points. A variation point is a location identifier in software at which the variation will occur. Software product line engineering (SPLE) manages the creation of systems' family. These systems are characterized by their features, i.e. logical units of behaviour specified by a set of functional and non-functional requirements [14]. Features may be common or vary between systems.

SPLs are described by variability models. In this thesis, we use the feature model that is a widespread tool to represent commonality and variability in SPL. Feature model was introduced in [44] to capture the problem space of a Software Product Lines (SPL), a family of software systems with some commonalities and significant variabilities. A feature model captures the potential variant features of members of an SPL in a tree structure, containing those features that are common to all members and those that vary from one member to the next. A particular member is defined by activating the desired nodes from the feature model [24]. This is known as a feature model configuration problem, i.e, a problem of selecting a subset of the features to optimally satisfy user's requirements. A node in a feature model is a logical unit of behaviour characterized by a set of functional and non-functional requirements [14]. A set of inter-feature relationships allows to specify (i) mandatory and optional parent-child feature relationships as well as (ii) alternative (XOR) and (OR) feature groups (see the legend in Figure 8). The parent-child feature relationships that comprise feature model constraints C are summarized as follows, where $C = \{C_1, C_2, C_3, C_4, C_5, C_6\}$:

- Mandatory-feature group: Given a parent feature fs_k and its mandatory-child feature fs_l,
  - C_1: If fs_k is selected, then its child feature fs_l must be selected.
- Optional-feature group: Given a parent feature fs_k and its optional-child feature fs_l,
  - C_2: If fs_k is selected, then its child feature fs_l can be selected or not.
- Or-feature group (IOR): Given a parent feature fs_k and its or-children features,
  - C_3: If fs_k is selected, then at least one of the children features must be selected.
- Alternative-feature group (XOR): Given a parent feature fs_k and alternative-children features,
  - C_4: If fs_k is selected, then only one of the children must be selected.
- Includes relationship: Given two features fs_k and fs_l where fs_k includes fs_l,
  - C_5: If fs_k is selected, then fs_l has to be selected as well.
- Excludes relationship: Given two features fs_k and fs_l wherefs_k excludes fs_l,
  - C_6: If fs_k is selected, then fs_l cannot be selected.

2.5 NSGA-II

Genetic algorithms (GAs) are widely used to solve complex service composition optimization problems as reported in the survey paper published in [41]. In this thesis, we particularly selected the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [27] that has shown better performance over other candidates (such as SPEA2) for multi-objective optimization problems for service composition [52].

NSGA-II starts with an initial population of random solutions called chromosomes. Each chromosome is composed of L genes and is encoded by a binary string. The population is sorted based on non-domination into a hierarchy of fronts. The non-dominant solution set in the population is placed in the first front. Solutions dominated by the first front are placed in the second front, and so on. Each solution s is assigned a non-domination rank s_{rank} equals to its front, rank 1 is the best. Since each solution is a set of objectives to be minimized, the non-domination can be defined as follows:

Definition: let s and t be two solutions where $s = \{obj_1^s, \ldots, obj_y^s\}$ and $t = \{obj_1^t, \ldots, obj_y^t\}$, we say that s is non-dominated by t, i.e., $s_{rank} < t_{rank}$, if $\forall obj^s \in s$ and $obj^t \in t, obj^s \leq obj^t$ and $\exists obj^s | obj^s < obj^t$.

Non-dominant solutions at the same level are sorted based on local crowding distance (s_{dis}) which measures how close each solution is to its neighbours. Large distance results in better diversity in the population. In other words, each solution s has two attributes to guide the selection process towards the optimal solutions: (1) Non-domination rank (s_{rank}), and (2) Local crowding distance (s_{dis}). The solution selection is defined as follows [27]:

Definition: Given two solutions s and t, we say that solution s is preferred over solution t, i.e., $s \prec_{\mathcal{N}} t$, if ($s_{rank} < t_{rank}$) or (($s_{rank} = t_{rank}$) and ($s_{dis} > t_{dis}$)).

Therefore, solutions are sorted in different fronts where the first one contains the non-dominated solutions (i.e. non-comparable solutions) while the second front contains solutions that are dominated by the solutions at the first front, but nondominated by the solutions at the third front and so on. Within the same front, we differentiate solutions (non-comparable w.r.t the order) based on the distance. As argued in [27], non-dominated solutions with the same crowding distance have no contribution to the convergence of the algorithm. Therefore, in such cases, a random solution is selected.

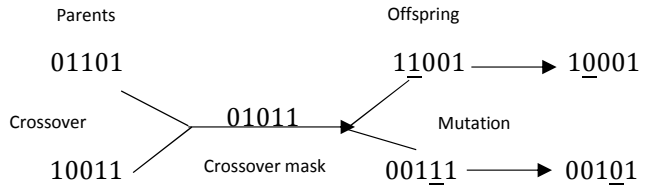


Figure 4: Crossover/Mutation operators

To produce the next generation, NSGA-II first creates a child population, called offspring, by applying binary tournament selection, crossover and mutation operators. Parents, solutions, are selected based on the rank and crowding distance. The crossover operator produces two offsprings by copying selected bits from each parent. We use the uniform crossover which employs a randomly generated mask to decide which parent's gene the offspring will inherit. The mutation operator randomly flips bits by choosing a single bit from an offspring and changing its value. Figure 4 illustrates an example of these operators. The crossover mask dictates which offspring inherit the bit from the parent. If the bit is 1 in the mask, the top offspring will inherit the bit from the top parent, if the bit is 0, the top offspring will inherit the bit from the bottom parent. The mutation operator randomly flips bits of the offspring. After creating the first population, the offsprings are sorted based on non-domination and the first M solutions are selected to form a new population, where M is the population size. Again, the selection is based on the rank and the crowding distance. The process continues to generate generations of offsprings to find better solutions.

2.6 Kalman Filter Model

Kalman filter model [74] estimates the state vector in a linear dynamic system, which is the nature of our online QoS prediction. The main steps of the Kalman filter algorithm are illustrated in Figure 5: (1) calculate the Kalman gain, (2) calculate/update the prediction values, and (3) update the prediction error. These steps are repeated to minimize the error.

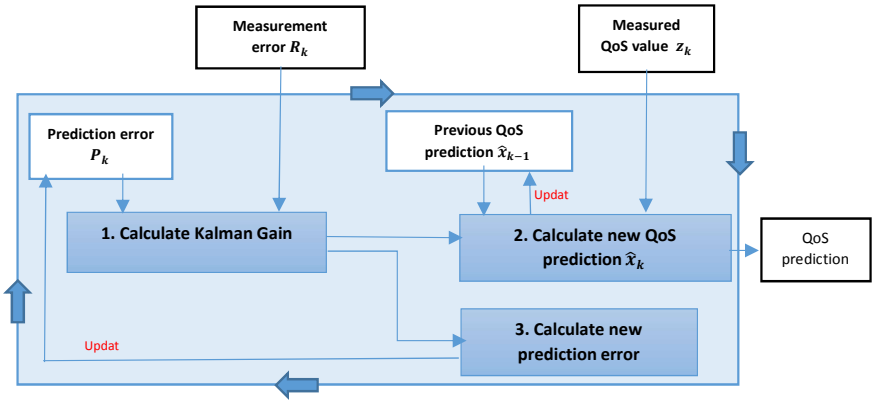


Figure 5: Kalman filter model

2.7 Dependency Network

Unlike the bayesian network, dependency network $\mathscr{D}$ is a graphical model that approximates the full joint probability distribution over the corresponding domain [36]. Moreover, the graphical structure of the dependency network is not required to be acyclic. Therefore, dependency networks can represent mutual dependencies or cycles among domain variables. The graphical structure of a $\mathscr{D}$ is a directed graph where each node represents a random variable X_i in the problem domain. Each random variable has an associated conditional probability distribution $P(X_i|X\backslash X_i) = P(X_i|\mathbf{pa_i})$ where $\mathbf{Pa_i}$ are the parents of X_i and $\mathbf{Pa_i} \subseteq (X_1,...,X_{i-1},X_{i+1},...X_n)$. Edges represent the global constraints among nodes and their absence entails the independence of the nodes.

For parameter learning, any probabilistic regression or classification model can be used to model the conditional probability distribution $P_i(X_i|\mathbf{pa_i})$. Parameters are estimated based on the maximum likelihood method that selects the set of parameters that maximize the probability of data given the model. This thesis adopts probability trees for modelling conditional probability distributions. A probability tree consists of internal nodes representing the binary tests, and leaves containing probability distributions for the target variable.

For structure learning, first, we lean the structure of each variable independently, then

the dependency network is constructed by linking dependent variables. To this end, for each variable, we search for the structure that optimizes the structure effectiveness score. Particularly, a greedy hill-climbing search driven by the scoring metric iteratively improves the current structure until the score does not increase.

Finally, through Gibbs sampling, the joint probability distribution of the dependency network is inferred. Gibbs sampling starts with a random initiation for each random variable X_i, and then in each Gibbs sweep iterates over the variables in a fixed order and resamples the value of each X_i from its local distribution $P_i(X_i|\mathbf{pa_i})$.

2.8 Relational Dependency Network (RDN)

Relational dependency networks (RDNs) [62] extend dependency networks used in relational databases. RDNs are graphical models that represent an approximation to the joint probability distribution over a relational data set and allow cyclic dependencies that are ubiquitous in relational domains.

2.8.1 Relational Database

A relational database consists of multiple related tables that represent relationships between classes of objects or their attributes. A unique identifier, the primary key, characterizes each entry in a table, whereas the foreign keys connect the tables. The relationships between tables could be one-to-one, one-to-many, or many-to-many relationships. For example, a one-to-many relationship corresponds to the connection of one row in a table to multiple rows in another table. To process relational data, it has to be flatten or propositionalized to obtain the data in an attribute-value format. This entails that data should comes in a single table with the attribute-value format where each row represents an instance and columns represent attributes of those instances. Despite the simplicity of this approach, it produces

propositional data with many attributes due to the complex relationships that might exist between objects, such as one-to-many and many-to-many relationships. In addition, representation of real-world relational data (e.g., service composition) in an attribute-value format leads to statistical bias in the results. Alternatively, first-order logic could be used to represent attributes of objects and relations between them by constructing relational features that capture the relational information. Then, a learning algorithm could be adopted to select the optimal features set that represents the problem domain.

2.8.2 RDN Representation

This thesis adopts RDN representation published in [69] using datalog (a subset of the first-order logic language that represents complex relational domains). The language consists of three types of symbols: constants, logical variables, and predicates. A constant is denoted with a lower-case letter to represent a specific object. A logical variable (logvar) is denoted with an upper-case letter. Logical variables represent placeholders for a specific subset of objects in the domain. Predicate represents the properties of objects or relations among objects. Each predicate P/n, where $n > 0$ is the arity of the predicate, has a finite range, $range(P)$. Unlike, traditional logic where the range of a predicate is in $\{false, true\}$, the range could be categorical or numeric. An atom is a predicate where each arity is either a constant or a logvar. A literal is an atom or its negation. A substitution maps each logvar to logvar or a constant while grounding substitution maps each logvar to a constant. In addition, rules can be used to define the content of a relational database that is not explicitly represented in the database. The rules can be represented as normal clauses of the form $H \leftarrow L1, L2, ..., Ln$ where H is an atom and $L1, L2, ..., Ln$ are literals. The tuples of the relational database are specified as facts which each is represented as a clause H with an empty body.

Datalog contains a set of random variable declarations (RVD) which defines the random variables in a domain. Random variables are represented by probabilistic predicates. Hence, random variables can take any value from the associated range to the probabilistic predicates. For example, the random variable declaration for an atom H is $random(H) \leftarrow L1, ..., Ln$ where $L1, ..., Ln$ is a conjunction of literals specifies that H is a random variable in the model if $L1, ..., Ln$ are true. A closed-world assumption, i.e. the groundings of predicates that are not specified explicitly in the interpretation are false, is used to guarantee the evaluation of the conjunction in the right-hand side of a random variable declaration.

2.8.3 RDN Learning

We use a set of random variable declarations to define the random variables in the domain. We also construct a space of relational features to learn random variable dependency. We learn dependency statements $(P|Parents(P))$, which define for each random variable P the other random variables that it depends on $Parents(P)$. In addition, we learn the conditional probability distributions (CPDs) to model the distribution of the target predicate P on the parent set of relational features $Parents(P)$. In other words, we independently learn a locally optimal CPD for each predicate. The parent set of P is determined by iterative searching for the optimal relational features that maximize the utility score (i.e. pseudo-loglikelihood). Parameters estimation method for the CPDs for the dependency statements depends on the range of the predicates. This thesis uses linear Gaussian CPD since the range of the predicates is continuous [47].

Finally, RDN model is obtained by conjoining all learned local distributions. Similar to dependency network, inference in RDN is be performed using an ordered pseudo-Gibbs sampler.

2.9 Context-aware Service Trust

This thesis adopts the concept of trust as defined in [83, 57, 33] being the confidence one has in the behaviour of others. Trust resources could be subjective and/or objective trust.

2.9.1 Subjective Trust

Subjective trust uses the direct interactions data between the truster and the trustee to estimate the service trust. We view subjective trust and context variables as follows:

- **Definition:** Subjective trust is the probability of providing satisfactory QoS values under context variables.
- **Context variables:** We consider price-awareness and profit-awareness variables as examples of context variables that impact the future behaviour of services and providers respectively.

2.9.2 Objective Trust

Objective trust uses feedback from service referees due to the lack of subjective trust resources. We view objective trust and context variables as follows:

- **Definition:** Objective trust is the reputation of a service measured using referees' feedback.
- **Context variables:** We consider collision attacks as examples of context variables.

2.9.3 Bootstrapping Trust

Newborn services have neither subjective nor objective trust resources, thus bootstrapping trust is used to initiate their trust values. We view bootstrapping trust and context variables as follows:

- **Definition:** Bootstrapping trust is the trust assessment of newly deployed services.
- **Context variables:** We consider white-wash attacks as examples of context variables.

2.10 Literature Review and Discussions

2.10.1 Adaptive Service Composition

An adaptation taxonomy is defined to facilitate the analysis of adaptation approaches [16]. This taxonomy has a set of dimensions that describe expected facets of adaptive service compositions, namely Why, When, What and How adaptation takes place.

Dimension: Why? This taxonomy dimension illustrates the goal behind the adaptation. The basic goal of adaptation is to allow the service to fulfill its functional and/or non-functional requirements. With regard to nonfunctional requirements, the goal of most existing approaches is to maintain and enhance the delivered QoS values.

Dimension: What? This taxonomy dimension refers to the level in which changes are carried out in order to achieve the adaptation goal. The level of adaptation can be classified into service level and process level.

The process-level reflects a high strategic level, while the service-level reflects a low tactic level. A process-level captures the whole picture of the business logic that will guide the high-level adaptation plan. Moreover, for an efficient adaptation with continuous business growth, the service-level is responsible for a smaller part of the business logic toward achieving the ambitions outlined in the high-level adaptation plan. Accordingly, a service-level adaptation is conducted first. If it is not successful, a process-level adaptation is planned.

If a composite service only uses service-level adaptation, it will crash in the case of an unsuccessful adaptation. By considering both adaptation levels, the composite service

can continue to function by reconfiguring its workflow. On the other hand, if a composite service only uses process-level adaptation, unnecessary complete process reconfigurations would be used for the simple replacement of failed services. Therefore, considering both levels reduces the adaptation time complexity. However, in the case of an unsuccessful service-level adaptation, the adaptation time complexity of considering both adaptation levels would be higher than if only a process-level adaptation would have been used. Nevertheless, this additional time required by the service-level adaptation is still negligible compared to the whole adaptation time. Most approaches address the adaptation problem either at the process-level only [2, 3, 15, 60] or at the service-level only [7, 35, 53, 66].

Dimension: How? This taxonomy dimension defines adaptation actions used to solve the adaptation problem. Based on adaptation levels, the adaptation process triggers the appropriate adaptation actions, such as service selection or workflow reconfiguration at the service level or process level adaptation, respectively. In [7, 17, 35], the authors focus on service selection actions to select new optimal constituent services, while workflow reconfiguration actions that modify the business process are studied in [2, 17]. In addition, this thesis proposes a new service-level adaptation action, namely constituent substitution to replace the failed constituents locally.

Dimension: When? This taxonomy dimension illustrates the time when adaptation is performed. Run-time adaptation can be distinguished from two perspectives, on-line and off-line [16]. On-line approaches use historical QoS data to predict SLA violation at run-time. Off-line approaches use historical QoS data to predict future SLA violations but do not predict at run-time. However, most of the aforementioned adaptation approaches, use QoS values in steps within the adaptation process to maximize the business value of the service. Thus, this thesis gives a deep look at on-line QoS prediction approaches that focus on maintaining the overall QoS values for the composite service. These approaches predict QoS values in order to detect SLA violations, which enable the adaptation managers to

Table 1: SLA violation prediction

approach	Off-line prediction	Online prediction	
		historical data	Context-aware
[98]	✓		
[99]		vast	
[82]		vast	
[51]		sample	
This thesis		sample	✓

trigger the right adaptation actions for the running composite services and prevent penalties.

State-of-the-art on-line QoS prediction approaches, rely on vast historical data to build the prediction model, which is inefficient for on-line QoS prediction. Moreover, they do not consider the dependency relations between QoS and context variables that come from the dynamicity and uncertainty of the running environment.

Table 1 compares our prediction model with existing approaches in SLA violation prediction. Table 2 summarizes the characteristics of the adaptation process in this thesis against other existing approaches, and Table 3 illustrates self-properties for composite services studied by this thesis against the state-of-art.

Conclusive Remarks The aforementioned adaptation approaches share the drawbacks of a centralized environment, making them inappropriate for composite services that run in distributed environments. However, distributed approaches are detrimental to the performance because of communication overhead. Moreover, these approaches use closed world assumptions, i.e. they define in advance the context changes. They lack support for analyzing the inherent variability of dynamic adaptation at design time to guide adaptation in an unpredictable open world.

Table 2: Characteristics of the adaption process

approach	Why	What		How			When	
	QoS	service	process	constituent substitution	constituent selection	Workflow reconfiguration	online	offline
[7]	✓	✓			✓			
[60]	✓		✓		✓			
[15]	✓		✓		✓			
[16]	✓	✓			✓			✓
[2]			✓			✓		
[53]	✓	✓			✓			
[17]	✓	✓	✓		✓	✓		✓
This thesis	✓	✓	✓	✓	✓	✓	✓	

Table 3: Self-adaptive capabilities

Approach	Self-healing	Self-optimizing	self-configuring	Self-protecting
[7]		✓		
[60]		✓	✓	
[15]		✓	✓	
[16]	✓	✓	✓	
[2]			✓	
[53]		✓		
[17]	✓	✓	✓	
This thesis	✓	✓	✓	✓

2.10.2 SLA Management

The authors in [65] draw attention to the problems of having uncontrolled dependencies between software modules and introduce the concept of information hiding. A pioneering work in [6] provides insights about dependency analysis solutions in software engineering domain.

Likewise, for composite services, the knowledge of service dependency is essential to avoid unpredictable SLA violations. The authors in [7] propose a correlation-aware service composition approach where QoS service dependencies are considered. The authors in [28] propose a QoS-aware service composition approach that considers QoS service dependency. The authors in [34] present a framework for modelling the QoS dependency relations using different orchestration pattern of a composition. Alike, The authors in [20] propose QoS dependency-aware service composition approach based on the Pareto set model. These approaches incorporate correlation aware methods to prune redundant services and reserve the services with QoS correlations that may be integrated into optimal composite services. In addition, the authors in [9] propose a probabilistic dependency graph to discover dynamic dependencies among services by analyzing service execution data. The authors in [91] analyze and capture dependencies between services in a composition in a semi-automatic manner in a dependency model. Based on this model, the authors

validate the SLA against the effects of events such as service failure or SLA renegotiation at run time. However, these approaches are based on service dependencies information described in the SLA. They are limited in the face of implicit dependencies in the dynamic running environment.

Therefore, a number of machine learning-based approaches have been proposed to work out predictions in real-time automatically after design. The authors in [50] propose a prediction model to predict SLA violation during runtime. A regression machine learning model is used for training data captured from historical process instances. The authors in [76] propose an approach for SLA violations prediction based on Naive Bayesian Classifier using measured datasets (QoS of used services). The authors in [67] propose a Bayesian network-based probabilistic QoS model to indicate the conditional independence relationships among QoS attributes.

Conclusive Remarks Those approaches cannot be scaled to real-world environments. In addition, they do not consider the different types of QoS dependencies, namely vertical, horizontal and cyclic QoS dependency. Moreover, they rely on propositional data which assumes services are independent. Thus, these approaches are in short supply for capturing service dependencies. We extend these approaches by taking the advantage of RDN to analyze different QoS dependency relations, since ignoring the dependency relations overestimates the computed values and this overshoot is detrimental to the stability of the prediction model [21, 58].

2.10.3 Feature Model-based Adaptive Service Composition

There are approaches that provide support for handling variabilities in composite services, which help choose the most appropriate variant at run-time [2, 3, 23, 90]. For example, in [2], the dynamic adaptation of the composition is supported by activating and deactivating feature nodes at the process-level, i.e. reconfiguring the composition logic. This approach

is based on the *closed world* assumption in which changes are known at design-time and there are predefined adaptation actions to deal with, which is not the case in the dynamic environment where composite services operate. In [3], a framework is proposed based on the *open world* assumption where changes are unknown at design-time. The framework is guided by general rule premises in a centralized knowledge base using forward chaining.

Conclusive Remarks These approaches share the drawbacks of the centralized approach such as high overhead. Moreover, they do not adequately consider QoS requirements in their adaptation algorithms, since they focus on QoS of the composite service while ignoring the QoS of the constituents.

2.10.4 Service Trust

A. Subjective Trust

The current computational trust models are based on feedback, statistics, fuzzy-logic, or data mining [80]. This thesis extends statistics-based subjective trust models to consider the dynamic environmental contexts, QoS metrics, and the dependency relationships between them. Particularly, we focus on bayesian network-based models usually deployed in the literature of related work.

In [87], a bayesian network approach is proposed to combine the trust of P2P-style interactions between agents. The authors demonstrate that the exchange of information about trust increases the performance of the network. In [63], a trust model based on the bayesian network is proposed to integrate both subjective and objective trust sources. Based on these sources, the final trust value is calculated. In [59], a bayesian network query is proposed to select service candidates in a composite service. In [57], the authors use a multinomial generalized Dirichlet distribution in learning bayesian networks to model QoS and compute QoS-based trust values. The authors learn and model the composition structure of composite services.

To deal with the context of the environment, [79] presents a framework to estimate QoS. Their framework involves three steps. It begins with building a bayesian network model to represent the QoS capabilities of the service. Then, the model is trained with feedback from different sources to learn the unknown parameters for the service. Finally, QoS is estimated by making probabilistic inferences on the basis of certain context variables. In [19], the authors propose a bayesian network-based trust approach to evaluate trust from user satisfaction experiences. The authors represent user satisfaction experience as a binary value, with 1 indicating satisfied and 0 not satisfied, which is the outcome of a Bernoulli trial. However, this work is based on the available social networks as prior knowledge, which is not granted. The focus of their work is SOA-based IoT. In [86], a context-aware approach for trust management of IoT service networks is proposed. The proposed model in this work predicts the trustworthiness of a service provider based on its behaviour in proving QoS in response to context variables.

Conclusive Remarks The scalability of bayesian network-based trust models is limited because the size of conditional probability tables grows exponentially with the number of parents of a node [37]. The aforementioned approaches disregard the cyclic relations that could link various QoS metrics and context variables. Consequently, the likelihood that QoS degradation would activate context variables has not been considered.

B. Objective Trust

Many models have been proposed to aggregate feedback provided by peers to estimate the trust. Usually, these proposals weight the feedback based on the credibility of the peers providing them. [22] uses the scores of the referees to weight the feedback. The authors average feedback to calculate the credibility score for a given trustee. [55] uses different assessment metrics, such as referee credibility and past feedback history to estimate the

trust. [85] aggregates the feedback using a Kalman aggregation method. Also, they use estimated feedback variance to detect malicious feedback and minimize their influence. [84] adopts the cumulative sum control chart to detect malicious feedback and prevent them using a Bloom filter. [92] uses referrals about common peers of the truster and each referee to estimate the credibility score to the referee. [58] proposes a personalized similarity-based credibility score. The truster, in their approach, asks for a recommendation request of a trustee which is certain about its trust. The similarities between the received feedback and the known trust define the credibility level of the referees. Moreover, [58] uses a cluster-based algorithm factoring the outliers to identify malicious referees, which are considered as anomalies. Clustering-based anomaly detection techniques assume that anomalous instances lie in sparse and small clusters [5]. Based on this assumption, [58] considers honest referees reside within a dense area as they have small variance relating to the same trustee whereas malicious referees have a larger variance and occupy a scattered area. Unlike these approaches, [25] focuses on detecting dishonest feedback rather than dishonest referees. They consider feedback as an outlier when it expresses a behaviour with a low level of occurrence in the data set.

Conclusive Remarks

The aforementioned approaches are limited in detecting the behaviour characteristics of the colluders described by [43]. We will discuss these characteristics in more detail in Section 4.1.2.

C. Bootstrapping Trust

[55] assigns trust values to new services based on the rate of maliciousness in the system. With a low rate of maliciousness, new services are assigned a high initial trust value, otherwise, they are assigned low trust. On the other hand, [93] observes the behaviours of new services throughout the testing time. Then, using Hidden Markov Models, they model the

observation sequence to detect the behaviour of the services. They compare the observed behaviour against predefined trust patterns to assign an initial trust value to the service.

Conclusive Remarks

The aforementioned approaches motivate malicious services to receive a low rate of maliciousness through a white-washing attack. Besides, in case the observed behaviour follows undefined patterns, the observation-based approach cannot bootstrap the trust.

Chapter 3

Service Adaptation Management

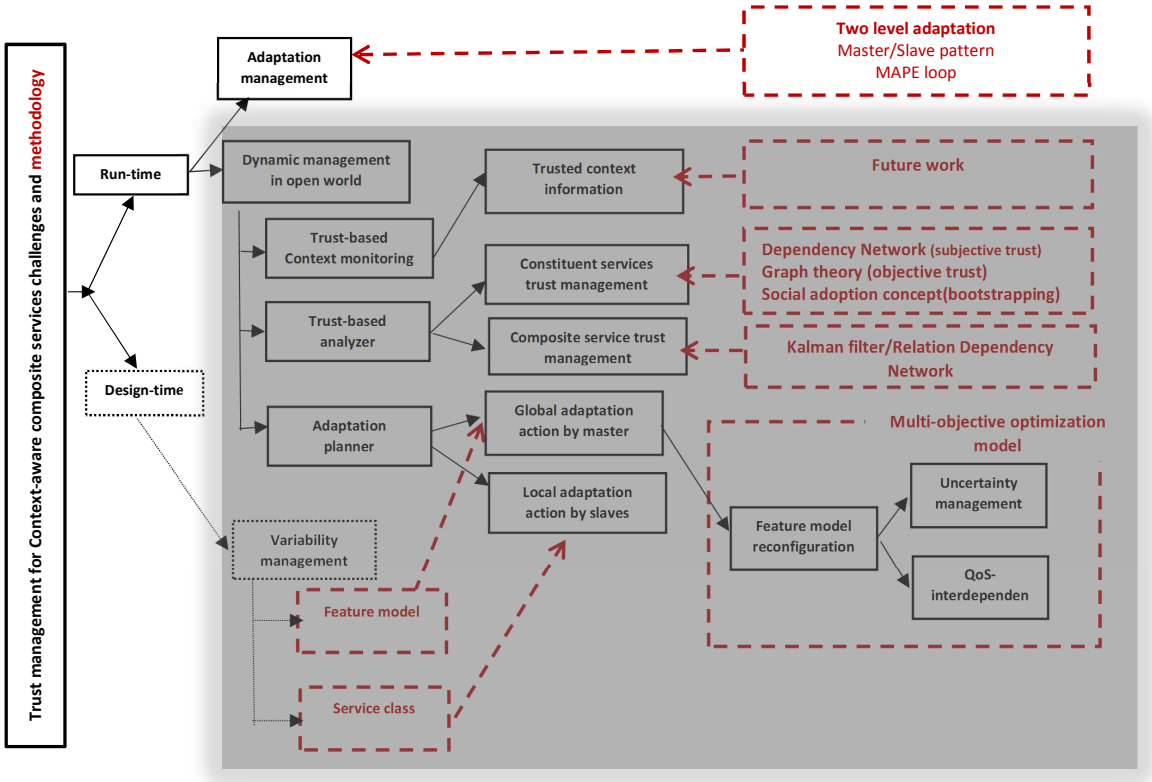


Figure 6: Chapter 3 challenges

Figure 6 (unshaded part of the figure) shows the addressed challenges in this chapter. Consequently, this chapter starts with an overview of the proposed approach in Section 3.1. Then, it highlights the main tasks of the master and the slaves managers in Section 3.2. Finally, it presents the experimental results and the conclusion in Section 3.3 and Section 3.4.

3.1 An Overview of the Proposed Approach

To describe the proposed approach, we will use an example of a composite service for order processing. Figure 7 details the operations of order processing using the Business Process Model Notation (BPMN) where the composition of internal components (i.e. Automatic order approval and Get items/warehouse) and external constituents (i.e. Payment calculator, PayPal payment, Email invoice, and Stander shipping) are illustrated. The process starts when an order is approved. Then, the order is checked for availability at the warehouse. If unavailable, the order will be discarded and the process will end. Otherwise, the total payment is calculated by a payment calculator service taking into account the available discounts. Then, the payment is processed by the PayPal service with a valid account. Finally, an e-mail will be sent to the user with the invoice and the order will be delivered by the standard Shipping service.

The designer of this composite service assumes a reliable and efficient execution, which is not the case in reality. This is due to the dynamic running environment. A composite service is subject to run-time context changes such as:

- Addition/removal of a task from the composite service.
- Failures in a service operation or service is performing below SLA constraints.
- Changing end-user QoS requirements.

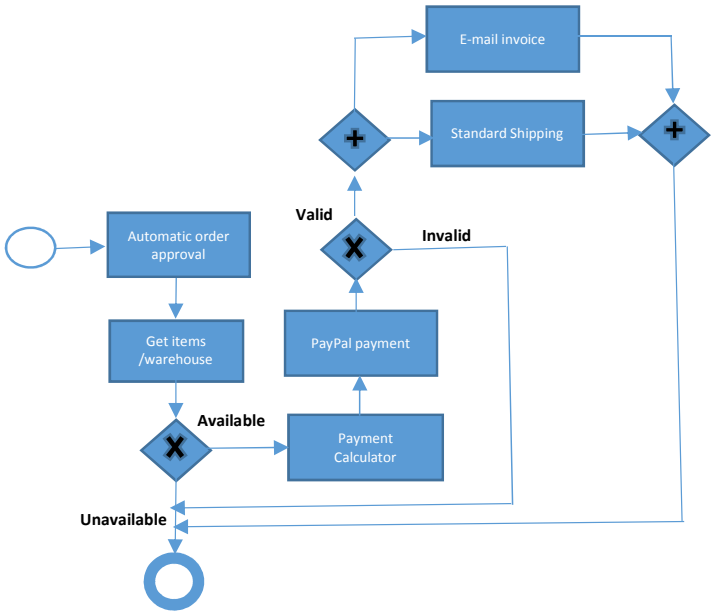


Figure 7: Order-processing BPMN

- Changing profit and/or price.

These changes may lead to SLA violation and penalty consequences. Examples of SLA constraints (i.e. SLOs)and penalties (Ps) are listed below:

- SLO1: The shipping time of order processing should be $\leqslant$ 5 days.
- SLO2: The availability value of the order processing should be $\geq$ 97%.
- SLO3: The cost of the order processing should be $\leqslant$ \$700.
- P1: The user should be entitled to 2% discount per each 1 day delay.
- P2: The user has access to 2% discount per each 10% decrease in availability.
- P3: The user would not pay for an additional cost.

Composite service operations that violate SLA constraints trigger adaptation actions, which are not cost-free. For example, the order processing service has a standard shipping constituent as shown in Figure 7. The SLA constraints of this constituent specify that the

shipping time is 3 days while its availability is 90% for a cost of $30. In some cases where the items are ordered from the warehouse due to lack of stock, SLA constraints for the composite service (e.g. SLO1) could get violated because of the additional time required to prepare these items. Replacing the standard shipping constituent with an express shipping constituent, whose shipping time is 1-day while its availability is 96% for a cost of $100, could enhance the performance, but entails an additional cost. This additional cost could be less than the paid penalties by the provider if he did not switch to express shipping (e.g. penalty for 1 day delay and 10% decrease in availability: 2% * $700 + 2% * $700 = $28). However, the SLA violation reduces the provider reputation and small penalty cases are not usual. Therefore, composite service adaptation is required to take place.

Figure 8 illustrates the feature model for the order-processing example. The figure shows the different possible variants of the order-processing service used to adapt the service upon a contextual change. The *Prepare-Order* feature has two mandatory child features that must be selected, *Prepare-item* and *Get-from-warehouse*, whereas selecting *Buy-item* is optional. Either *coupon* or *apply discount* must be selected in the alternative feature group if their parent feature is selected, while at least one feature (e.g., *express or standard*) must be selected in an *or* feature group if their parent feature is selected. Accordingly, an order-processing composite service could be built by activating some feature nodes while deactivating the others as it is shown in Figure 8. A change at run-time, such as delay in shipping time, will require deactivating the *standard* and *Buy-item* nodes and activating the *express* node.

Many possible feature model configurations (i.e., a set of activated features that satisfy the feature model constraints discussed in Section 2.4) can exist in a feature model. Industrial feature models consist of hundreds of features, making feature configuration a complex task [11]. Furthermore, different features have different impacts on the QoS properties of composite service. For example, activating *express* shipping decreases the shipping time,

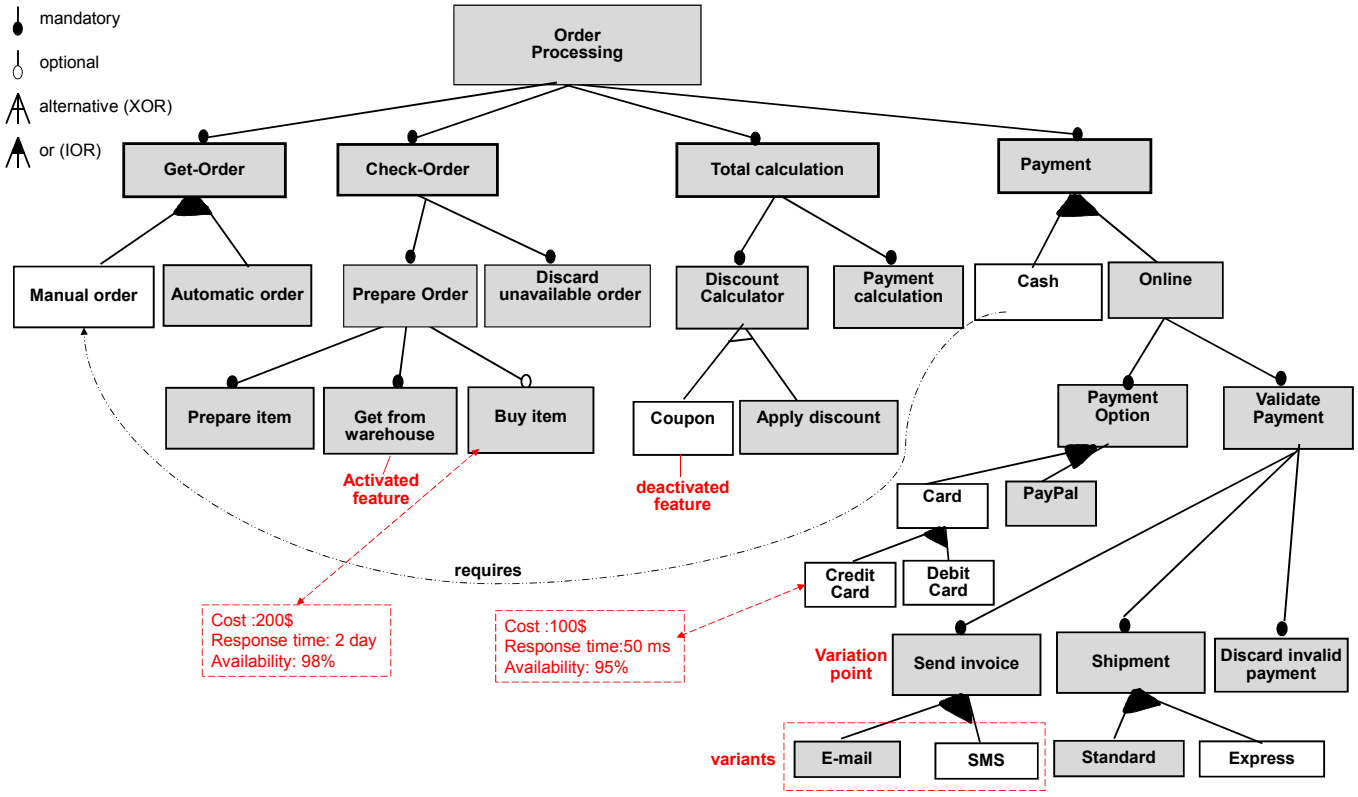


Figure 8: Order-processing feature model

but increases the cost, and activating *Buy-item* increases the availability at the expense of the shipping time. Considering different QoS metrics when selecting features is a complex task because of the interdependencies between these metrics, i.e., a change in the value of one metric could result in a change in a different one. Figure 9 illustrates this where minus signs refer to inverse relationships while plus signs indicate positive relationships. Furthermore, we can have conflicting user's requirements in the feature model, such as requesting high performance and low cost.

In consequence, the main goal of this thesis is to design context-aware composite services that are able to keep satisfying QoS requirements when encountering changes in dynamic and uncertain environments. To achieve this, we design the decentralized architecture of our approach based on the MAPE loop reference model and master/slaves pattern as discussed earlier in Chapter 1.

At design time, we analyze the inherent variability of the composite service to include

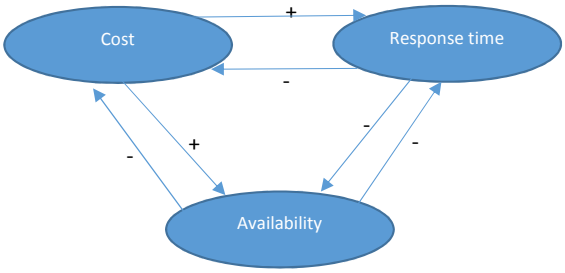


Figure 9: QoS-interdependence

the variants of the service using the feature model to guide run-time adaptation. Indeed, we annotate feature nodes with QoS metrics as shown in Figure 8. The standard feature model does not consider weights in its features. However, some proposals have extended the feature model notation with non-functional metrics [10]. Likewise, we extended the feature model-based service adaptation to represent the QoS metrics of each feature. Since our approach selects the optimal feature subset, which is the closest to satisfying the SLA constraints based on the QoS metrics of each feature, these metrics are considered implicitly as the feature weights. For example, the features of credit card and debit card are associated each with specific shipping time and availability, which implicitly form the weights of these two features. Our approach will select the best feature among these two that satisfies the SLA based on their associated weights.

Moreover, we model the adaptation problem as a multi-objective optimization, which makes our solution practical for the open world with no need for predefined adaption actions. In other words, when a change is predicted, the workflow of composite service is reconfigured using the feature model restricted by the constraints without predefined rules. In addition, we propose distributed local adaptation actions by the slaves for replacing failed constituent services promptly, which reduce the time complexity of reconfiguring the composite service. To this end, each node of the feature model is realized by a service class i.e., a set of services with the same functionality but with different QoS metrics [12]. This enables the slaves to have alternative backups for the failed constituents.

At run-time, the master and slaves go through the different MAPE steps to manage

the service adaptation. In the monitor step, sensing mechanisms are used to capture QoS values and contexts, i.e. situational information about the run-time environment. Each slave monitors the associated constituent service to ensure that it satisfies its own QoS constraints. The global monitoring of the composite service is performed by the master to secure end-to-end QoS, i.e. SLA.

In the analyze step, the master and slaves examine the incoming QoS information about the composite service and the constituents, respectively. Besides, they consider the context of the running environments to predict violations in QoS and/or SLA constraints. This supports their decisions involving adaptation actions.

During the plan step, the master and slaves decide on the adaptation actions for the composite service and the constituents, respectively. Whereas the execute step safely deploys the adaptation actions that are produced by the planners.

This thesis focuses on the analyze and plan steps. In particular, we focus on predicting QoS and/or SLA violations and composite service adaptation in a dynamic open world. When the SLA is predicted to be violated, the master adapts the composite service by reconfiguring the workflow logic. Similarly, when QoS constraints of a constituent service are predicted to be violated, the slave substitutes the failed service with its backup from the associated service class. Such prompt substitution maintains the global performance of the composite service since it reduces the need to composite service adaptation that is the most time-consuming step in the adaptation process.

The proposed adaptation approach promotes self-adaptive capabilities of the composite service through: (1) Self-configuring capability, which enhances the ability of the service to face open world changes; (2) Self-optimizing capability by adopting the master/slaves pattern, which allows for monitoring QoS performance and triggering promptly adaptation actions; (3) Self-healing capability via the adaptation actions that allow the service to discover, diagnose and react to context changes; and (4) Self-protecting capability by the

on-line prediction models that detect malicious performance.

When the SLA is predicted to be violated, the master adapts the composite service by reconfiguring the workflow logic, i.e. activating and/or deactivating feature model nodes. Particularly, the master selects the optimal set of feature model nodes that maximizes the end-to-end QoS performance and minimizes the cost under SLA and feature model constraints. We consider the annotating QoS metrics with each feature node as a local SLA for this node. Thus, our approach chooses the feature configuration that yields the closest aggregated SLA, aggregated from local ones, to the SLA using a multi-objective optimization approach. Different aggregation models can be defined for each workflow connector, such as XOR, AND, and OR [39]. However, these models can be reduced to the sequential aggregation model as shown in [40]. Accordingly, the aggregation functions indicated in our approach focus on the sequential model, as illustrated in Equations 25 and 26 which will be explained in Chapter 5. Afterward, for each activated feature node to achieve a certain task, a slave selects the optimal implementing constituent that delivers the QoS values specified in the local SLA.

Similarly, when a local SLA is predicted to be violated, a slave substitutes the failed service with its backup from the associated service class. Such prompt substitution maintains the global performance of the composite service. Composite service adaptation is the most time-consuming step in the adaptation process, but thanks to the proposed local service adaptation action, the need for global adaptation has been reduced and the overall performance is improved.

3.2 Master/Slaves Managers

The inadequacy of centralized adaptation approaches for composite services are (1) inefficient adaptation for scale-free environments, and (2) inflexible and ineffective adaptation

towards changes in the environments. This is due to the distributed environments of composite services. On the other hand, distributed adaptation approaches produce communication overhead, which is detrimental to performance. Therefore, this thesis proposes a decentralized dynamic adaptation architecture based on master/slaves pattern as discussed earlier in Chapter 1.

The master/slaves pattern enables us to manage services adaptation in scalable settings. Unlike centralized approaches in which a single entity is responsible for the whole environment, in our approach, slaves locally monitor and adapt distributed constituent services and the master monitors the composite service to be reconfigured in order to provide the agreed QoS constraints. Such an architecture can overcome the problem of the centralized approach by distributing the adaptation management load over several nodes (i.e., master and slaves). Fortunately, the architecture enables the slaves to substitute failed constituents promptly to maintain the overall performance and avoid the global adaptation which has high computational complexity.

The hierarchical nature of the master-slaves architecture facilitates the communication between the salves and the master. The slaves communicate with the master to reconfigure the workflow when replacing a failed service is unsuccessful. Moreover, the master interacts with the slaves after the reconfiguration to select the implemented services. Since each slave is controlled by a local SLA and the master is controlled by the overall SLA, i.e. the aggregation of the local ones, there is no conflict among the slaves and the master.

3.2.1 Master Manager

The master manager goes through the MAPE loop to control the composite service. It monitors the overall performance and adapts the service to provide the expected end-to-end QoS.

In the analyze step, the master predicates SLA violation to support its decision-making

in the plan step regarding the adaption action for the composite service. The proposed SLA violation prediction approach will be discussed in Chapter 4.

In the plan step, the master adapts the service at the process-level by reconfiguring the service workflow. We propose a solution based on the feature model in Chapter 5.

3.2.2 Slaves Managers

Slaves locally monitor and adapt the distributed constituent services through the MAPE loop. Each slave is responsible for the local adaptation of a constituent service at the service-level.

In the analyze step, each slave estimates the service trust to predict QoS constraints violation under the context of the running environment. We propose a dependency network trust model that enables slaves to capture the dependency relations among QoS metrics and context variables to estimate the likelihood that the services will provide acceptable QoS metrics considering the current context. This step will be discussed in Chapter 4.

In the plan step, slave selects the optimal implementing constituent that delivers the QoS values specified in the local SLA for each activated feature node to achieve a certain task. This step comes after workflow reconfiguration for the composite service by the master. Besides, this thesis proposes adaptation actions for constituent service substitution that enable slaves to promptly replace the failed services. Thus, the self-adaptive capabilities of the composite service are enhanced. This step will be discussed in Chapter 5

3.3 Experiments

All the experiments have been conducted on a 1.1 GHz Intel Core 2 Duo laptop with 8 GB of RAM. We use the CloudHarmony dataset [1] of 53 different services operating in

[1] https://cloudharmony.com/

different parts of the world during a period of a whole month and the average availability is recorded. Those services are owned by well-known providers such as Amazon Web Services and Agile Cloud. To make our experiments fair with the centralized benchmark adaptation approach proposed by [17], which uses a small number of services, we have selected a subset with the same number from the used dataset.

3.3.1 Distributed Environment

This experiment explores the effectiveness of our adaptive approach in a distributed environment. Table 4 compares the QoS values and the cost obtained by our approach against those obtained by the centralized benchmark adaptation approach. To ensure a fair comparison, we use the same initial configuration of the benchmark approach for the response time, availability, and cost requirements. The two typical settings are [1, 0, 1] for config1, and [0, 1, 0] for config2, where 1 denotes a higher preference of the corresponding requirement over the one with 0.

As shown in Table 4, our approach outperforms the centralized approach [17] by reducing the response time and the cost by 50%, and enhancing the availability by 2%. These improvements are important for on-line adaption approaches that run in real-time.

Table 4: Comparison between our approach and the approach presented in [17] in a distributed environment

QoS	config1		config2	
	our approach	centralized approach	our approach	centralized capproach
Response time(ms)	368	952	286	797
Availability (%)	98.32	96.87	98.59	96.86
Cost($)	1.56	2.94	1.21	2.53

The improvements are since our approach monitors the QoS values of the composite

service for triggering adaptation actions upon violation. Since we use a distributed architecture in which each slave monitors each service, we are able to capture every single change in the environment. The centralized approach uses a single entity for monitoring the whole environment, which prevents it from capturing detailed information from each constituent affecting its overall performance in terms of response time, availability and cost. Finally, in our approach, the master optimizes the configuration of the composite service to respond to a global change.

In contrast to the centralized benchmark approach, our proposal considers QoS attributes in addition to the cost w.r.t resource consumption and penalties in the optimization problem to reconfigure the workflow (as we will discuss in Chapter 5). This guarantees an optimal reconfiguration while minimizing the total cost and not exceeding the cost constraint which supports the satisfaction of both the service user and provider. However, SLA monitoring by the master and slaves in our approach entails additional costs.

3.3.2 Scalability and Robustness

This experiment explores the scalability of our approach to large-scale environments. First, we increase the size of the feature model from 75 to 200 features. Accordingly, the workflow size increases from 28 to 115 service classes, each service class containing 100 services. Figure 10 compares the performance overhead caused by the growth of the environment in our approach against the centralized benchmark approach. As shown in the figure, our approach can scale well with the growth of the feature model. We can conclude that our proposal can be effectively applied to a large-scale environment as it reduces the execution time by 15% compared to the centralized benchmark approach. Our approach achieves better scalability because the adaptation process is carried by multiple entities, namely the master and slaves, whereas this process is carried by a single entity in the benchmark approach, which usually fails in large-scale settings for its incapability to deal with increasing

the number of constituents. However, the scalability of our approach is constrained by the networking capacity of the master. The master's capacity could saturate since it would be overwhelmed by the slaves' messages.

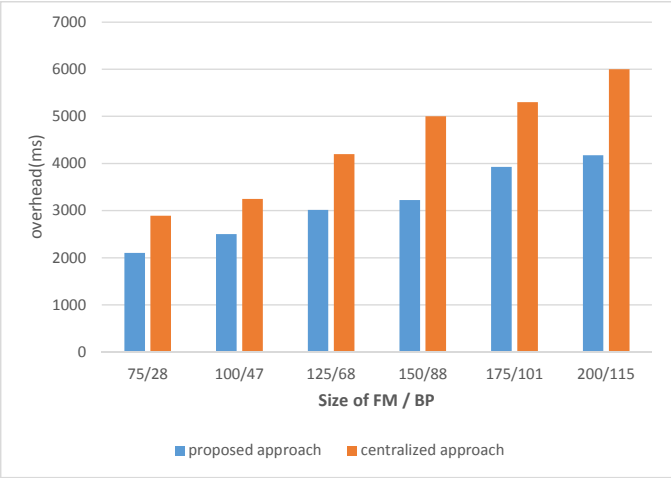


Figure 10: The performance overhead in a growing environment

Furthermore, we conducted a set of experiments while changing the service network scale. In particular, we changed the number of services from 500 to 10000 while fixing the workflow to 28 service Classes. Figure 11 shows the relationship between the service network scale and execution time. The trend line shows a polynomial relationship between these two factors, which proves the effectiveness of our approach in large-scale environments. In contrast to the service selection approaches, e.g. [17], where all service classes are considered at the same time by the same entity, which is a complex task, the proposed local adaptation action enables each slave to deal with only one class to select the implemented service. Thus, our approach has the ability to substitute the failed constituents instantly to maintain the robustness of the composite service. However, the failure of local adaptation action to find the required substitute necessitates composite service adaptation action to reconfigure the service. Thus, unlike service level adaptation approaches, the robustness of the service comes at the expense of the execution time.

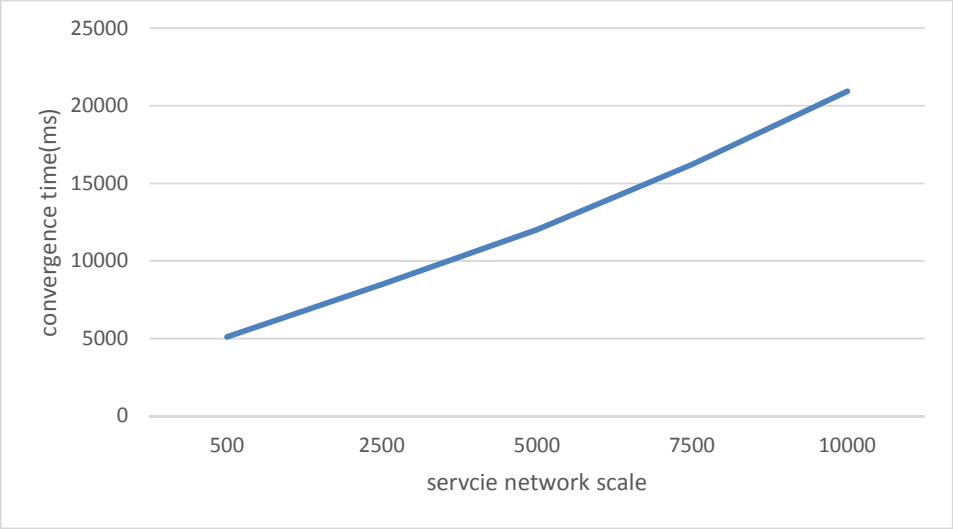


Figure 11: The execution time in a growing environment

3.4 Conclusion

In this chapter, we proposed a decentralized approach for composite services adaptation by applying the master/slaves pattern. Unlike centralized approaches in which a single entity is responsible for the whole environment, in our approach, slaves locally monitor and adapt distributed constituent services and the master monitors and reconfigures the workflow of composite service in order to deliver the requested QoS. This enhances the overall performance of the composite service in terms of response time, availability and cost in dynamic large-scale environments. Besides, the hierarchical nature of the master-slaves architecture facilitates the communication between the Salves and the master, which avoids performance detriment because of communication overhead in distributed environments. Thus, this chapter achieved the first research objective (Objective 1) discussed earlier in Chapter 1, which is aimed at proposing an adaption architecture that fits the distribution nature of composite services without being detrimental to the performance by communication overhead.

Master/slaves mangers go through MAPE loop steps to manage composite services

running in a dynamic and uncertain world. This thesis focuses on the analyze and plan steps. Particularly, we investigate run-time QoS violation prediction models for prompt service adaptation to maintain and secure the service performance. Chapter 4 illustrates the proposed prediction models in Section 4.1 and Section 4.2 that enable, respectively, the slaves and master to analyze QoS and context information to predict any performance violation. Chapter 5 disuses the adaption actions to be taken by the master and slaves upon violation prediction in Section 5.1 and Section 5.2.

Chapter 4

Service Trust Management

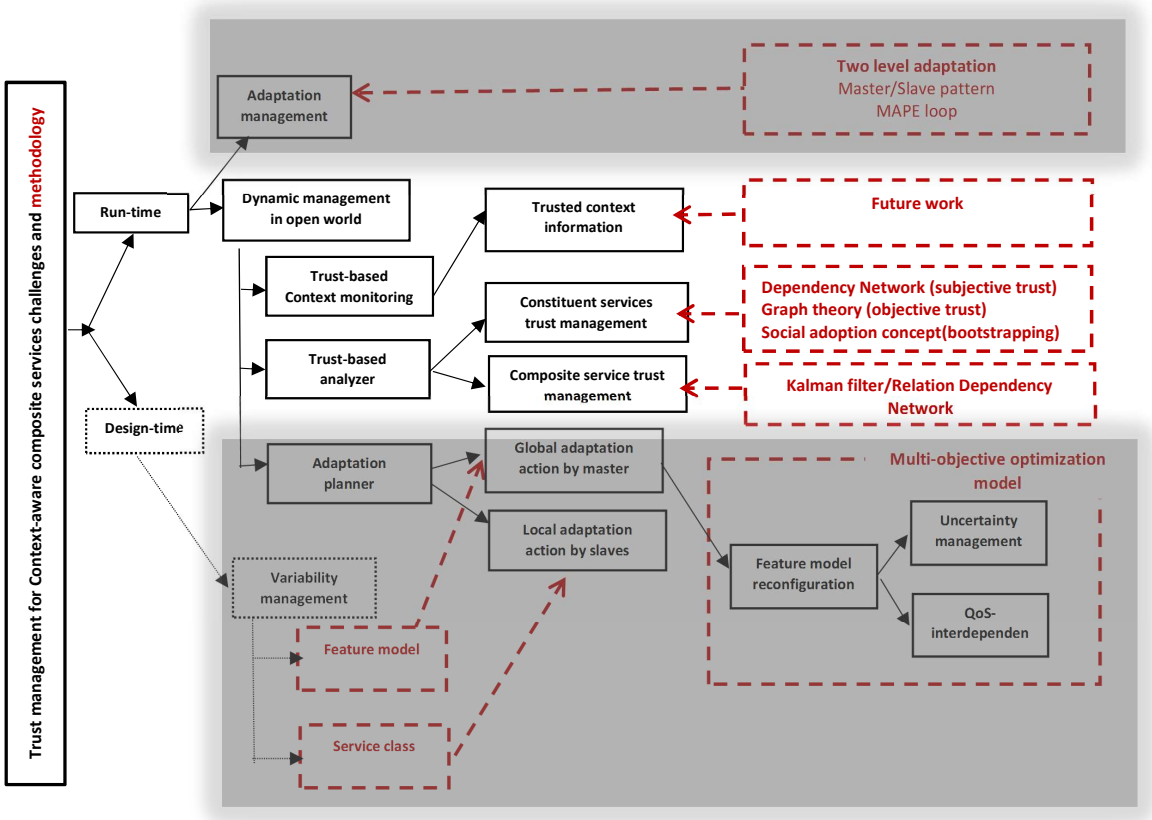


Figure 12: Chapter 4 challenges

Figure 12 (unshaded part of the figure) shows the addressed challenges in this chapter. Consequently, this chapter presents our proposed solutions that tackle the challenges facing trust-based managers, namely the master and slaves running at the analyze step of MAPE loop. In Section 4.1, we present a context-aware multi-dimensional trust management model that empowers each slave responsible for a constituent service to estimate the service trust to predict QoS violation. In Section 4.2, we present a context-aware QoS prediction approach that enables the master responsible for the composite service management to predict SLA violation. Then, in Section 4.3, we present the experiments. Finally, Section 4.4 concludes this chapter.

4.1 Multi-Dimensional Trust

The continuous dynamic environment is one of the challenges that trustworthy services management faces. Services in such an environmental context have difficulty securing an acceptable quality of service (QoS). However, a few research attention has been paid to the comprehensive trust management for context-aware services as reported in the survey paper published in [72]. Therefore, this section proposes a trust management framework that establishes service trust by considering the direct trust from the truster (subjective trust), aggregating referrals about the service in a collusion-resistant manner (objective trust), and bootstrapping new services. We introduce a subjective trust model based on the formalism of dependency networks to dynamically predict the provided QoS in response to context environment changes. The proposed approach leverages the dependency relations that exist among the QoS metrics and environmental context variables. The novelty at the subjective trust level lies in considering the dynamic cyclic dependency relations that enhances the prediction accuracy. However, subjective trust based on direct interactions could be insufficient to make the trust estimate credible. Hence, on top of the subjective layer, we propose an objective trust management model resilient to collusion attacks by leveraging the power

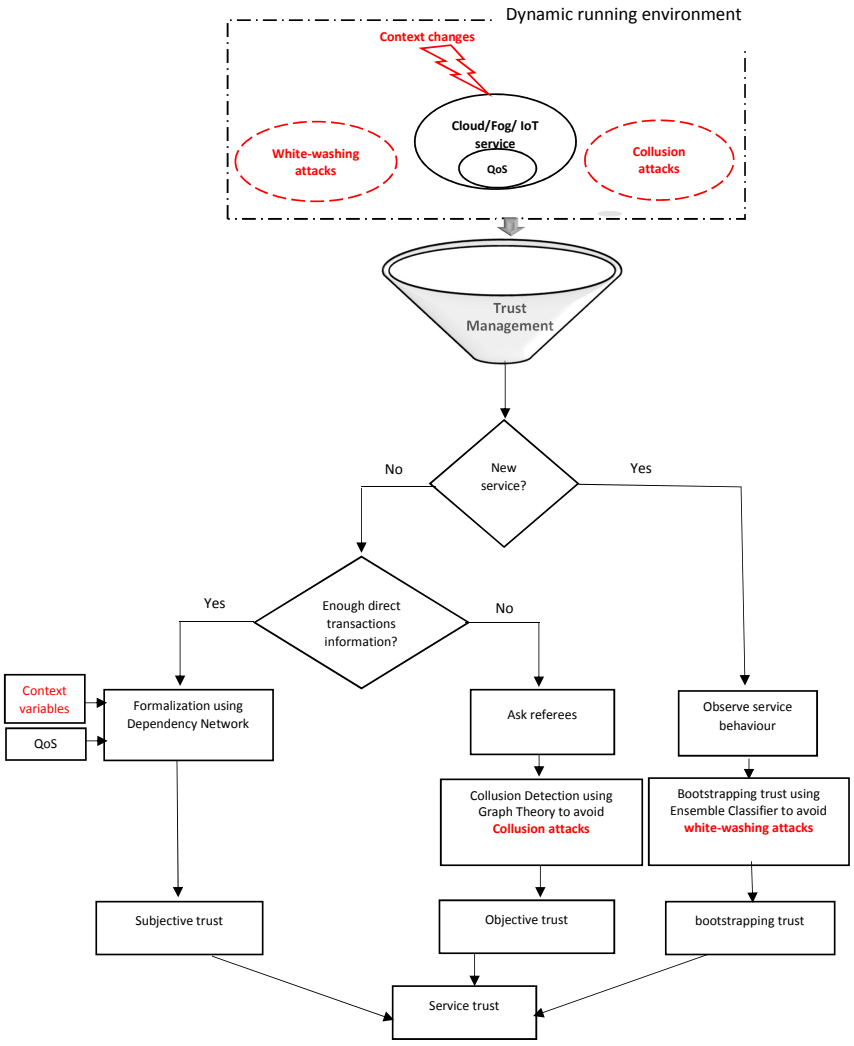


Figure 13: Multi-dimensional trust for services

of mass collaboration among referees. Finally, we propose a bootstrapping mechanism that is resilient to the white-washing attacks (i.e. services with poor trust reset to start afresh with new identities) by observing the behaviours of newcomer services with no trust resources using the concept of social adoption to estimate their initial trust values. Such attacks can significantly affect the delivered performance of trust-based services, accordingly, the proposed approach is concerned to avoid them.

Unlike most of the existing trust frameworks which focus solely on one particular trust issue, our approach establishes the service trust starting from subjective service trust considering the environmental context and the cyclic dependency, aggregating referrals in a

collusion-resistant manner, and bootstrapping new services. Figure 13 exhibits the research problem we are addressing in this section. The figure also displays the research framework and identifies the used techniques and tools, namely dynamic dependency networks, graph theory and ensemble machine learning classification.

4.1.1 Subjective Trust

The current computational trust models are based on feedback, statistics, fuzzy-logic, or data mining [80]. We extend statistics-based subjective trust models to consider the dynamic environmental contexts and QoS metrics in addition to the dependency relationships among them. Most existing subjective trust models, particularly Bayesian network-based models [59, 57], predict the service trust based on QoS metrics while ignoring the running environment. Thus, these approaches are limited in trust modelling. Furthermore, the scalability of Bayesian network-based trust models is limited since the size of conditional probability tables grows exponentially with the number of parents of a node [37]. On the other hand, the authors in [86, 19, 79] take in their consideration the running environment, i.e. the context of the environment. However, they disregard the cyclic relations that could link various QoS metrics and context variables. Consequently, the likelihood that QoS degradation would activate context variables has not been considered.

This thesis investigates the problem of subjective trust for context-aware services. Subjective service trust is defined as the likelihood of providing a requested service that satisfies the agreed QoS while considering the context variables. Particularly, we predict the QoS given the dynamic environmental contexts to measure the subjective trust which relays on the direct interaction among the services. We consider the context as the state of the running environment and the context variable as the environmental condition that affects the behaviour of the service [48, 61, 86]. The price-awareness that impacts the future

behaviour of a service is a concrete example of a context variable. Hence, we view the subjective trust of context-aware services from the angle of the likelihood that these services will provide acceptable QoS metrics considering the current context variables. The authors in [58] argue that ignoring the dependency relations overestimates the effective trust. This overestimation is counterproductive to the stability of the trust model [21].

Therefore, we extend the calculation of the service trust to take into account the cyclic dependency relations that occur between the values of QoS and the context variables of the running environment. To illustrate these cyclic dependency relations, we consider two awareness-driven variables for the context: price and profit. The price is the context variable from the side of the user, while profit is the variable from the side of the provider. QoS degradation will adjust the behaviour of the users driven by the price-awareness to pay less [96]. The profit-driven provider, on the other hand, adjusts the delivered QoS based on the offered prices, which results in degrading the QoS delivered at low prices [96]. Thus, the two awareness variables: price and profit exhibit a cyclic dependency relation as the activation of one variable affects the delivered QoS, which in turns affects the other variable.

Therefore, we aim to develop a trust model that dynamically predicts the probability of delivering an acceptable QoS. To this end, we capitalize on the formalism of dependency networks. Similarly to the Bayesian network formalism which is a graphical model, the formalism of dependency networks computes the joint probability distribution. However, it utilizes Gibbs sampling to approximate the probability distribution [36]. Unlike Bayesian networks, the graphical structure of dependency networks may contain cycles. Thus, dependency networks are able to capture the mutual dependencies and cycles among variables of the problem domain. The nodes of a dependency network graph, which is a directed graph, represent these variables where each node contains the local conditional probability of the corresponding variable given its parents in the network. Whereas the dependence and the independence of the nodes are represented by the precedence and the

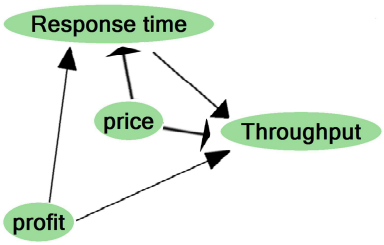


Figure 14: Bayesian network of a service trust

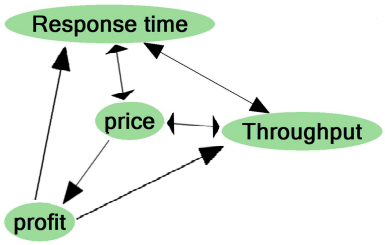


Figure 15: Dependency network of a service trust

absence of the edges respectively.

Moreover, The authors in [37] argue that dependency networks are easier to learn from complete data (i.e., data with no missing values), compared to Bayesian networks, thanks to the approximate method described in [36]. In addition, the dependency network is more robust due to the fact that any scalable classification or regression algorithm can estimate the local distribution of each node in the network.

Using the WS-DREAM dataset [71], we create two networks for each service that plays the role of the trustee: a dependency network and a Bayesian network. Each service is being managed by a slave playing the role of the truster. Unlike the Bayesian network, the dependency network is capable of capturing and representing the cyclic relations that exist among QoS metrics and context variables. For example, Figure 14 and Figure 15 depict respectively the Bayesian and dependency networks of a service using the WinMine Toolkit[1]. The presence of cyclic relations indicates that deterioration of QoS may result in some context variables being activated. Such cyclic dependency relations affect the service trust calculations. Ignoring such cyclic relations may lead to overestimated trust. For this reason, we consider in this Chapter the different dependency relations linking context variables to QoS metrics which have not been dealt with in the current approaches. Thus, the framework of dependency networks is being used instead of the formalism of Bayesian

[1] `https://www.microsoft.com/en-us/research/publication/the-winmine-toolkit-2/`

networks usually deployed in the literature of related work.

Dependency Network Learning

In the remainder of this thesis, an upper case token (e.g. X_i) denotes a variable, a lower case token denotes its value (e.g. x_i), a bold-face capitalized token (e.g. $\mathbf{Pa_i}$) is used to indicate a set, and its instantiation is represented by a bold-face lower-case token (e.g. $\mathbf{pa_i}$). We use calligraphic token for templates and graphs (e.g. $\mathscr{G}$).

We define a service using a set of random variables $X = (X_1, \cdots, X_n)$. Each variable represents either a QoS metric or a context variable. For instance, let X_{RT} and X_{TP} be two QoS metrics representing respectively response time and throughput of a given service, and let X_{PC} and X_{PF} be two context variables denoting respectively price and profit of the same service. Formally, the service would be represented as follows: $X = (X_{RT}, X_{TP}, X_{PC}, X_{PF})$.

The corresponding dependency network $\mathscr{D}$ for X is $(\mathscr{G}, \mathscr{P})$ where $\mathscr{G} = (V, E)$ is a cyclic directed graph. Each node V_{X_i} represents a random variable $X_i \in X$, and each edge $e_{i,j}$ connects V_{X_j} with V_{X_i} iff X_i is one of the parents of X_j. $\mathscr{P}$ is the set of local conditional probability distributions, expressly $\mathscr{P} = (P(X_1|X \backslash X_1), ..., P(X_n|X \backslash X_n))$

$\mathscr{D}$ is constructed by learning the local conditional distribution of each $X_i \in X$. Probabilistic decision trees are adopted to estimate local conditional distribution of each X_i, i.e. $P(X_i|X \backslash X_i) = P(X_i|\mathbf{Pa_i})$ where $\mathbf{Pa_i}$ are the parents of X_i and $\mathbf{Pa_i} \subseteq (X_1, ..., X_{i-1}, X_{i+1}, ...X_n)$. In other words, for each target variable X_i, we estimate $P(X_i|\mathbf{Pa_i})$ given the remaining variables. Then, we adopt a score-based structure learning approach to search and evaluate the space of candidate structures to select the optimal one. Thus, learning a decision-tree structure for each X_i is an optimization problem for finding a structure $\mathscr{T}$ with the maximum score given observed data D. Particularly, we optimize the pseudo-loglikelihood (PLL) of an assignment x to random variables X. Pseudolikelihood is an approximation to the joint

probability distribution of a collection of random variables. It estimates the set of conditional distributions independently rather than jointly. PLL that estimates the contribution for each relevant random variable conditioning on all other assignment values is calculated as follows:

$$PLL(x) = \sum_{i}^{n} log[P(X_i = x_i|\mathbf{Pa_i})] \tag{1}$$

In consequence, a greedy search algorithm is adopted to search the structure space and find the one with the highest PLL. It starts with $\mathscr{T}$, and for each modified structure $\mathscr{T}'$, by *edge deletions, edge additions, and/or reversals*, a new score is calculated. $\mathscr{T}'$ is accepted if PLL($\mathscr{T}'$) > PLL($\mathscr{T}$). This is repeated until no increase in the score of the tree is possible. Afterwards, the overall $\mathscr{D}$ of the service is constructed by linking each X_i with $\mathbf{Pa_i}$.

For probabilistic trust inference of the service, Gibbs sampling is adopted to recover the joint distribution of X, i.e. $P(X)$.

- We begin with some initial value $X^{(k)}$.
- The next sample $X^{(k+1)} = (x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)})$ is obtained by sampling each $x_i^{(k+1)} \in X^{(k+1)}$ by updating it according to the distribution specified by

$$p(x_i^{(k+1)}|x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)})$$

as follows:

$$X^{(k+1)} = \begin{cases} x_1^{(k+1)} \sim p(x_1|x_2^{(k)}, x_3^{(k)}, ..., x_n^{(k)}) \\ x_2^{(k+1)} \sim p(x_2|x_1^{(k+1)}, x_3^{(k)}, ..., x_n^{(k)}) \\ x_3^{(k+1)} \sim p(x_3|x_1^{(k+1)}, x_2^{(k+1)}, ..., x_n^{(k)}) \\ \\ x_n^{(k+1)} \sim p(x_n|x_1^{(k+1)}, x_2^{(k+1)}, ..., x_{n-1}^{(k+1)}) \end{cases} \tag{2}$$

- Repeat N times.

For particular instances, e.g. x_2 , we can compute $P(x_1|x_2)$ by fixing the value of X_2 to x_2 during the iterations. The advantage of this is the ability to answer probabilistic queries of the form $P(x_1|x_2)$, where $X_1, X_2 \subset X$. For example, we can infer the probability of getting a response time less than one second given throughput rate greater than 10 kbps as $p(X_{RT} < 1|X_{TP} > 10)$.

Accordingly, the trust of a truster u toward the trustee X is computed as follows:

$$trust_u(X) = \frac{1}{N} \sum_{k=1}^{N} W^k \times F(X^{(k)}) \tag{3}$$

where $F(X^{(k)}) = \sum_{i=1}^{n} p_i \times x_i^k$, s.t. p_i is a user-defined preference regarding the different used metrics. In addition, according to the law of large numbers, we define an ascending weight for the samples as follows:

$$W^k = \frac{k}{N-k+1} \tag{4}$$

Adapted Dependency Network

However, dependency network assumes that the relations linking QoS metrics to context variables are static, i.e., they are not changeable over time. Indeed, this assumption is not realistic since the dynamic running environment is influenced by different context variables. Additionally, the user's QoS requirements are volatile and changeable with time. For example, the time context variable has a different effect on other context variables and QoS values at peak service hours than non peak ones. At peak hours, services have higher pricing rates and service users must receive an acceptable level of QoS values. In such a manner, the dependency network representing the service trust has links and/or nodes changing over time. Accordingly, the service trust, the value to predict, is a time-sensitive

variable.

To incorporate this evolving nature, we extend the dependency network approach to consider dynamic environments. Particularly, the prediction model is adapted to deal with the concept drift, i.e., changing underlying relationships in the data. To this end, we modify the probabilistic decision trees to consider such a phenomenon. We are inspired by the algorithm called "Concept-adapting Very Fast Decision Trees (CVFDT)" proposed in [38] to compare the score of the tree at different times to find more efficient split attributes (binary tests) to adapt the model to the new data. This approach keeps the model consistent with the data without retraining it from scratch by updating the sufficient statistics at the nodes. However, the approach is proactive with no explicit strategy to detect the concept drift. Thus, the approach suffers from slow reaction to the concept drift in data. Therefore, we adopt the concept drift detector based on Fisher's exact test proposed in [26]. The Fisher's exact test works efficiently with small sample sizes, which makes it an applicable detector in highly dynamic environments, unlike the most statistical tests that require large sample size for precise results. It analyses the prediction results over two windows to track changes in the error rate of the model. A recent window containing the latest data instances and an older window are used for this purpose. The authors in [26] assign the windows size to 30 instances. As a new data instance arrives, it is added to the recent window, and the data is shifted to forget the oldest data instance in the old window.

The main idea behind this detector is that the model keeps its accuracy over the two windows, whereas a significant decrease within the accuracy over the recent window signals for a drift. Based on a $2x2$ contingency table that contains the errors and hits of two windows, the probability of all observed frequencies p is calculated, which should be greater than an adopted significance level, otherwise the null hypothesis is rejected indicating to a drift. The null hypothesis assumes that errors are equally distributed over windows. The authors in [64] suggest the significance level of the warning level and the drift level as w=0.05

Algorithm 1 adaptive local conditional distribution

```
1: Let 𝒯 be the set of the constructing tress of 𝒟
2: Let ALT(X) be the set of alternate trees rooted at node X
3: Let NEX be the set of new data instances in the recent data window
4: Output: 𝒯 with adapted local conditional distributions
5: Begin
6:     for each tree t ∈ 𝒯 do
7:         for each node X do
8:             for each alternate tree in ALT(X) do
9:                 use NEX to calculate its score
10:            end for
11:            select the alternate tree that maximizes the heuristic score
12:        end for
13:    end for
14: End
```

and d=0.003 respectively. Within the warring period, the internal nodes are scanned to adapt their local conditional distribution by adapting the underlying trees inspiring by the approach in [38] .

Algorithm 1 illustrates the adaptive local conditional distribution of each node. First, the dependency network is decomposed into the set of the constructing trees. For each tree, we scan the internal nodes to find more efficient splitting attributes to enhance the heuristic score capitalizing on the sufficient statistics. To this end, at each internal node, we compare the sub tree rooted at this node with all alternate trees rooted at it to select the one maximizing the heuristic score similarly to [38]. Afterwards, the overall $\mathscr{D}$ of the service is constructed by connecting the nodes to their parents.

4.1.2 Objective Trust

We extend the trust framework by leveraging objective trust resources to overcome the lack of subjective trust resources for a trustee. To this end, referrals toward the trustee are aggregated to compute the objective trust in a collusion-resistant manner. Since malicious referees collude to either promote or demote services, collusion attacks mislead trust

results.

Recent approaches to service objective trust evaluation capitalize on social networks for referees discovery to estimate the service objective trust [77]. However, this depends on the availability of friends who have direct interactions with the trustee, a condition that may be difficult to meet under the same context environment. In addition, the social discovery process has high system overhead.

To tackle these problems, we propose a novel service objective trust evaluation technique. The concept of the proposed technique is inspired by the open-source code projects, e.g. Linux, where software developers around the world collaborate to raise the level of quality. The advantage of this mechanism stems from the power of mass collaboration to establish a meaningful service trust. In such a manner, the truster initiates a request to a trusted third party. The request contains the direct trust of the truster u toward the trustee X under a context environment, $trust_u(X)$. The truster uses a hash function to prevent the referees from knowing the value of $trust_u(X)$. On the other hand, each referee r sends a feedback report $\{trust_r(X), profile_r\}$, where $trust_r(X)$ is feedback value toward the same trustee under the same context environment and $profile_r$ is the referee's profile which contains the history of his interactions. The third-party updates his central database with the referees' interactions history to detect collusion behaviours. Particularity, collusion among referees are detected to filter out the malicious colluders' reports as discussed in the next subsection. Then, the third party calculates the similarity between each non-malicious referee and the truster [19]. Finally, it computes the service trust $trust(X)$ toward the trustee by combining subjective trust and objective trust using a weighted average. The weight of each feedback $trust_r(X)$ is the similarity degree between this feedback and the truster own opinion $trust_u(X)$. Thus, the highest weight (i.e., 1) is given to the truster own opinion

$trust_u(X)$. The service trust is calculated as follows:

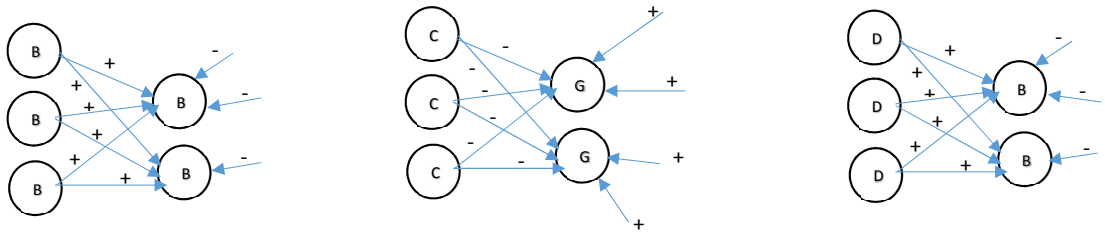
$$trust(X) = \frac{\sum_{r=1}^{RT}(1-|trust_u(X)-trust_r(X)|)trust_r(X)+trust_u(X)}{\sum_{r=1}^{RT}(1-|trust_u(X)-trust_r(X)|)+1} \quad (5)$$

where RT is the number of non-malicious referees and $1-|trust_u(X)-trust_r(X)|$ is the similarity degree between $trust_u(X)$ and $trust_r(X)$ which belongs to $[0,1]$.

Collusion Detection

Current approaches calculate service objective trust by assigning weights to referees' referrals according to their global credibility. However, these approaches do not address collusion in the calculation directly. Moreover, existing collusion detection solutions are mostly machine learning-based approaches to find frequent patterns that require that all observations be identically and independently distributed [88]. Therefore, they are still vulnerable to sophisticated collaborative attacks, i.e. collusion attacks.

In this thesis, we directly detect collusion based on referees history according to the threat model introduced in [43]. Figure 16 illustrates the behaviour characteristics of the colluders. Type A (not shown in the figure), represents colluders that are independently malicious and do not form any group. On the other hand, types B, C, and D form a malicious collective. Type B malicious colluders boost the trust of each other. However, we only consider the situations where type B malicious colluders boost the trust of the newcomers of type B in the malicious collective as Figure 16a shows. Whereas, Figure 16b illustrates that type C malicious colluders provide dishonest feedback to good services (G) which receive high ratings from others outside the colluding collective. Type D malicious colluders boost the trust of type B colluders which receive low ratings from others as it is demonstrated by Figure 16c. Particularly colluders overlap to boost or decrease the trust values. The visualized threat model motivates us to view collusion detection problem as a citation-based



(a) Threat model B (b) Threat model C (c) Threat model D

Figure 16: Visualized Threat model

clustering problem where we are interested in identifying overlapping referees to detect the malicious collective.

To this end, we propose a graph-based approach that is more suitable for collusion detection because it can capture the underlying relationships among referees via their feedback toward services. We view referees and services as nodes V of a directed graph $\mathscr{R} = (V, E)$ where an edge from a node p to a node q means a review from p toward q does exist. Moreover, the out-degree of a node p is the number of nodes it has a review for, and the in-degree of p is the number of nodes that have reviews for this node. We directly detect collusion based on referees' history of following colluder patterns illustrated in Figure 16.

For overlapping referees detection, we use the approach proposed in [46] that aims to find the highly authoritative nodes and hub nodes and detects the overlap in the sets of hub nodes pointing to the authoritative nodes. Authoritative nodes are those that have large in-degree while hub nodes have large out-degree to multiple relevant authoritative nodes. In our graph-based collusion detection modelling process, we have considered authoritative nodes as services that malicious referees (hubs nodes) are reviewing. Particularly, malicious referees boost or lessen the trust values of services by collaboratively reviewing the services which, in terms of graph theory, increases the out-degree and in-degree of the

nodes corresponding to the malicious referees and the services respectively. Therefore, we are interested in detecting the overlap in the sets of hub nodes (i.e., malicious referees) pointing to the authoritative nodes (i.e., services to be reviewed) to detect the collective collusion attacks. Each node $node$ is associated with two different weights, namely authority weight ($auth^{node}$) and a hub weight (hub^{node}). These weights are calculated as follows:

$$auth^{node} = \sum_{node':(node',node)\in E} hub^{node'} \times w^{(node,node')} \tag{6}$$

$$hub^{node} = \sum_{node':(node,node')\in E} auth^{node'} \times w^{(node,node')} \tag{7}$$

where we define $w^{(node,node')}$ as percent of positive/negative feedback in all feedback from $node$ toward $node'$ with regard to negative/positive feedback from all other nodes.

The set of authority weights $\{auth^{node}\}$ is encoded as a vector $auth$ with a coordinate for each node in $\mathscr{R}$. In a similar way, the set of hub weights $\{hub^{node}\}$ is encoded as a vector hub. These vectors are initialized by value 1. Then, the vectors are updated iteratively by Equations 6 and 7. We select the nodes with large coordinates in $auth$ and hub as authorities and hubs respectively.

To detect type D malicious and type B malicious colluders who boost the trust of type B malicious colluders that receive low feedback from other referees outside of the colluding collective, we define $w^{(node,node')}$ as the percentage of positive feedback in all feedback from $node$ toward $node'$ with regard to negative feedback from all other nodes as follows:

$$w^{(node,node')} = \frac{FD^{+}_{(node,node')}}{FD_{(node,node')}} + \frac{FD^{-}_{(V\setminus node,node')}}{FD_{(V\setminus node,node')}} \tag{8}$$

where $FD^{+}_{(node,node')}$ is the number of positive feedback from $node$ for $node'$, $FD_{(node,node')}$ is the number of feedback from $node$ for $node'$, $FD^{-}_{(V\setminus node,node')}$ is the number of negative feedback from all nodes except $node$ for $node'$, and $FD_{(V\setminus node,node')}$ is the number of

feedback from all nodes except $node$ for $node'$.

To detect type C colluders, $w^{(node,node')}$ is defined as the percentage of negative feedback in all feedback from $node$ toward $node'$ with regard to positive feedback from all other nodes as follows:

$$w^{(node,node')} = \frac{FD^{-}_{(node,node')}}{FD_{(node,node')}} + \frac{FD^{+}_{(V\setminus node,node')}}{FD_{(V\setminus node,node)}} \tag{9}$$

where $FD^{-}_{(node,node')}$ is the number of negative feedback from $node$ for $node'$, $FD_{(node,node')}$ is the number of feedback from $node$ for $node'$, $FD^{+}_{(V\setminus node,node')}$ is the number of positive feedback from all nodes except $node$ for $node'$, and $FD_{(V\setminus node,node')}$ is the number of feedback from all nodes except $node$ for $node'$.

4.1.3 Bootstrapping Trust

The newcomer services that have no interaction history to estimate their initial trust values are more likely to be overlooked in future service transactions for the lack of trust evidence. However, services with poor history could rejoin the service community with new identities to reset the past (i.e., white-washing attacks). Thus, one of the crucial issues in establishing service trust is trust bootstrapping, i.e. the assignment of initial trust values to newcomer services, that is a resilient to the white-washing attacks

Most of the existing bootstrapping approaches, for example, the approaches proposed in [81] and [18], are based on the existence of feedback, which is not always a realistic assumption. Moreover, bootstrapping approaches that use default values or punishments have high-performance overhead due to the communication and aggregation processes [81]. The default-value approach assigns default trust values for newcomer services [81]. A high default value of the trust will favour recently lunched services over existing ones that endeavoured to achieve a good trust value. However, newcomer services with low

default trust value will fail to be involved in transactions. Thus, either the already existing services or the newly added are favoured. This approach motivates malicious services with bad history to re-publish new identities to start over, known as a white-washing attack. The punishing approach overcomes white-washing by assigning low initial values of trust for the fresh services. On the other hand, fresh services are disadvantaged considering the low chance of making transactions and gaining trust. The adaptive approach assigns trust values for newcomer services based on the rate of maliciousness in the system [55]. However, the downside of this approach is that it is vulnerable to a white-washing attack. To overcome these limitations, this thesis estimates initial trust values for newcomer services through behavior observation, which contributes to our proposed bootstrapping mechanism by making it resilient to the white-washing attacks.

We propose a novel bootstrapping approach, based on the concept of social adoption where newcomers are adopted by trusted services, i.e. bootstrappers, to initiate their trust values. The motivations for trusted services to adopt newcomer services are: (1) the cost of new services is free/low-cost; and (2) the newcomer services would reduce the workload of trusted services. The incentive of newcomer services to behave well is the trust gain. The proposed approach counts on the observed values of QoS metrics of the newcomers under different contexts to estimate the service trust. Newcomer services will be monitored by the bootstrapper for a predefined time window. Similar to the approach proposed by [55], the newcomer services have no knowledge of the time of the evaluation period. During this time window, newcomer services will be monitored in different context environments. This mechanism makes our approach resilient to white-washing attack.

We use incremental classifiers for bootstrapping. Incremental classifiers are widely used for their efficiency and capability to learn from a stream of observations which makes them suitable for the bootstrapping problem. Unlike traditional classification techniques such as decision trees or neural networks that require large labelled training data which is

hardly obtained, the incremental classifiers, for example, naive Bayes classifiers, are initiated with a sample training dataset while the observed data is continuously used to extend it. Particularly, we consider instance-incremental learning where the classifier learns from each observation as it arrives [94, 49]. Accordingly, an incremental naive Bayes classifier is used by the bootstrapper. The initial sample dataset is obtained from the bootstrapper's history records of similar services to the newcomer, based on their functional and non-functional specifications. Each service $X = (x_1, \cdots, x_n)$ has a class label $c_l \in C$ and $C = (c_1, \cdots, c_L)$. In our case, we have two class labels, namely *trustworthy* and *malicious*. On top of the training dataset TD, the naive Bayes classifier, namely multinomial naive Bayes classifier with Laplace smoothing equal to 1 for zero probability cases, learns patterns of data, i.e. finds out the mapping relation between each new observation X and a class label c_l as follows:

$$P(c_l|x_1, \cdots, x_n) = \frac{P(x_1, \cdots, x_n|c_l)P(c_l)}{P(x_1, \cdots, x_n)} \tag{10}$$

After labeling each observation $X < X^*, c_l^* >, X^* = (x_1^*, \cdots, x_n^*)$, where $*$ denotes new observation, the classifier updates its parameters as follows [31]:

$$P(c_l) := \begin{cases} \frac{\delta}{1+\delta} P(c_l) & \text{if} \quad c_l^* \neq c_l, \\ \frac{\delta}{1+\delta} P(c_l) + \frac{1}{1+\delta} & \text{if} \quad c_l^* = c_l \end{cases} \tag{11}$$

where $\delta = |C| + |TD|$, s.t. $|C|$ is the number of class labels and $|TD|$ is the length of the training dataset.

$$P(X_i = x_i|c_l) := \begin{cases} \frac{\xi}{1+\xi} P(X_i = x_i|c_l) & \text{if} \quad c_l^* = c_l, x_i^* \neq x_i, \\ \frac{\xi}{1+\xi} P(X_i = x_i|c_l) + \frac{\xi}{1+\xi} & \text{if} \quad c_l^* = c_l, x_i^* = x_i, \\ P(X_i = x_i|c_l) & \text{if} \quad c_l^* \neq c_l \end{cases} \tag{12}$$

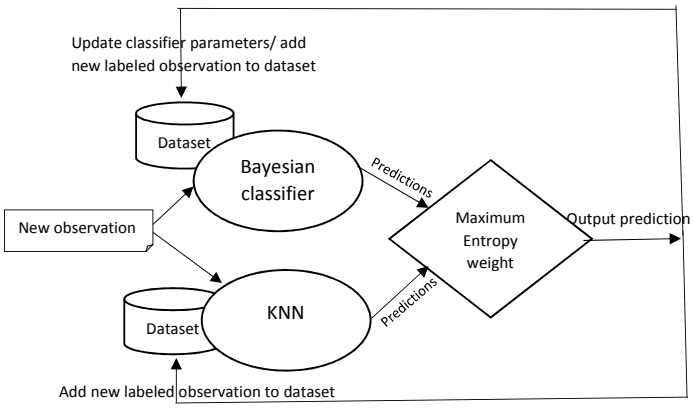


Figure 17: Ensemble classifier architecture

where $\xi = |X_i| + \text{count}(c_l)$, s.t. $|X_i|$ is the number of values for X_i and count(c_l) represents the number of samples with class label c_l.

The incremental naive Bayes classifier performs poorly under a small number of samples. However, as the number of samples increases, the performance of the classifier increases. This means, the early stage of the bootstrapping mechanism has a poor performance and produces noisy data that harm the overall performance of the classifier.

To solve this, we use an ensemble of classifiers which has been proven to be more accurate than a single classifier. Particularly, we combine the incremental Bayesian classifier with the K-nearest neighbor (KNN) algorithm as Figure 17 illustrates. In the case of the KNN algorithm, the situation is quite the opposite. KNN performs well with a small dataset which enhances the performance of the early stage of the classification. Thus, we combine the advantages of both classifiers to cover their drawbacks. KNN classifies a new observation by computing the majority of votes of the labels for K-nearest neighbor set (K). This set is determined by computing the distance, i.e. similarity, of the training data to the new observation using the following equation, then selecting the K nearest neighbor:

$$\arg\max_b \sum_{(x,c_l)\in K} (b = c_l)$$

where b is the majority voted class label. However, the standard KNN does not estimate

the probabilities of class labels [56]. To this end, we simply estimate the conditional probability $P(c_l|X) = Q/|K|$ where Q is the number of instances belonging to class c_l in the neighborhood set with length $|K|$.

An ensemble of classifiers arises the issue of how to combine the predictions. The most common technique for combining the prediction of classifiers is voting. In voting (majority vote and weighted majority vote), each classifier gives a vote for a prediction to obtain the final prediction which receives the most votes. Unlike the majority vote, the weighted majority vote relaxes the assumption that classifiers are equally accurate, which is more practical. However, most of the existing weighted majority vote algorithms are static where the weights of the combined classifiers are evaluated in the training phase. Therefore, they are not able to capture the changes in the stream of observations. On the other hand, dynamic weighted majority voting has the ability to deal with steam of observations to emphasize the contribution made by accurate classifiers and suppress the influence of others.

Current dynamic weighted majority vote approaches use genetic algorithms [75]. The problem of computing the weights is modelled as an optimization problem to select either the features to be used by a single classifier or the actual set of classifiers that have to be combined. None of the existing approaches are based on the statistical performance of the classifier. Therefore, we apply entropy and probability distribution to reflect the certainty of the classifier in the predicted class label, thereby, reducing the interference of unreliable noisy information and achieving accurate performance. Particularly, high entropy and a highly spread probability distribution indicate low certainty and vice versa. First, we compute the class probability distribution predicted by each classifier $P_v(c_l|x_1,\cdots,x_n)\forall c_l \in C$ where $v \in \{1,2\}$. Then, we compute the entropy of the class probability distribution H_v predicted by each classifier as:

$$H_v = -\sum_{l=1}^{L} P_v(c_l|x_1,\cdots,x_n)\log_2 P_v(c_l|x_1,\cdots,x_n).$$

Finally, we compute the entropy weight for each classifier as:

$$w_v = \frac{1-H_v}{\sum_{v=1}^{2} 1-H_v}.$$

The output prediction will be the prediction of the classifier with maximum weight. The new observation will be added with the predicted class label to training datasets as shown by Figure 17. As depicted in the figure, there is a circular structure between the output prediction and the input of two datasets. The arrows indicate that the output prediction is included to update the knowledge about the newcomer service which is under observation. This provides enough information to be able to assign an initial trust value to this new service.

4.2 SLA Violation Prediction

The master analyzes QoS information of composite service to forecast SLA violations. Based on the forecast, adaptation actions are invoked to prevent potential violations. To this end, we propose an online prediction approach based on Kalman filters that is effiecint in dynamic environments in the following subsection.

However, the online prediction approach fails to comprehensively consider service dependency that exists in the real business world at run time, which may lead to high economic compensation caused by breach penalties. Therefore, we propose an SLA violation prediction approach capitalizing on relational dependency network (RDN) to express and reason with service dependencies in a relational setting. We discuss this in details in Subsection 4.2.2.

4.2.1 Kalman Filter-based Approach

An online QoS prediction approach requires the following properties: (1) *efficiency*: the approach should not require large historical QoS data processing in order to reduce the time

required to make decisions. (2) *effectiveness*: the approach should be able to successfully predict as many SLA violations as possible since false predictions are costly and time-consuming. (3) *robustness*: the approach should be aware of continuous changes in QoS to adapt the composite service as early as possible as late adaptation brings additional cost. To meet these requirements, we propose an on-line QoS prediction model based on the Kalman filter algorithm [74].

The Kalman filter model estimates the state vector in a linear dynamic system, which is the nature of our online QoS predication. The main reason for choosing Kalman filters is because they do not require large historical data, making them convenient for online prediction. In comparison, time series models (e.g. ARIMA/GARCH) and neural network models (e.g. RNN) require large historical datasets to train the model which affects negatively the efficiency requirement. Moreover, offline trained neural network models are inefficient for online adaptation because of their incompetence in dynamic environments [68]. Another related model that works on small datasets is the Grey model [42]. However, we particularly chose the Kalman filter model over the Grey model for its ability to improve prediction accuracy on data series that may oscillate frequently in the short-term [89]. This would fulfill the effectiveness and robustness requirements.

The main steps of the Kalman filter algorithm are illustrated in Figure 5 in Section 2.6. The algorithm calculates the Kalman gain, calculates/updates the prediction values, and updates the prediction error. To reduce the estimation uncertainty, i.e., error, the Kalman filter algorithm recursively performs prediction followed by an update. It assumes that the state at a time k is evolved from the prior state at time $k-1$.

In equation format, given a state vector x_k that represents QoS metrics, namely response time, availability and cost, the QoS state dynamics at time k is described as follows:

$$x_k = Fx_{k-1} + w_k \tag{13}$$

where F is the state transition matrix which applies the effect of each state parameter at time $k-1$ on the state at time k and w_k is the process noise vector. The covariance matrix for w_k is Q.

In addition, the measured QoS vector is represented in terms of the states to be comparable to the predicted one using:

$$z_k = Hx_k + v_k \tag{14}$$

where z_k is the measured QoS vector, H is the transformation matrix to map the state vector parameters into the measurement domain, and v_k corresponds to the measurement noise vectors. The covariance matrix for v_k vector is R. In our case, we set $F_k = H_k = I$, where I is the identity matrix.

In the prediction step, the state vector is predicted from the state dynamic equation using:

$$\hat{x}_k = F\hat{x}_{k-1} \tag{15}$$

where $\hat{x}_k$ is the predicted state vector at the time window k, and $\hat{x}_{k-1}$ is the previous predicted state vector at the time window $k-1$. The covariance matrix of the predicted state is calculated by:

$$P_k = FP_{k-1}F^T + Q \tag{16}$$

where P_{k-1} is the previous predicted state error covariance matrix.

In the update step, the Kalman gain K_k matrix is calculated by Equation 17. Then, the predicted QoS state vector x_k at time k is updated by Equation 18. Finally, prediction error P_k is updated by Equation 19.

$$K_k = P_k(P_k + R_k)^{-1} \tag{17}$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - \hat{x}_{k-1}) \tag{18}$$

$$P_k = (I - K_k)P_{k-1} \tag{19}$$

This process will be repeated to reduce the error P_k.

Improper choice of the covariance matrixes of process noise (i.e., Q) and measurement noise (i.e., R) may significantly degrade the prediction performance. However, they are mostly determined using an ad-hoc procedure, in which Q and R are assumed to be constant during the estimation, which could not capture the dynamicity of the running environment. Therefore, we propose a tuning approach based on GA for estimating the optimal values of Q and R. Particularly, we find Q that minimizes the mean square error of current estimated QoS values against the previous ones by Equation 20 and R that minimizes the mean square error of the measured QoS values against the real ones by Equation 21.

$$\min \sum_{t=1}^{k} (\hat{x}_t - \hat{x}_{t-1})^2 \tag{20}$$

$$\min \sum_{t=1}^{k} (z_t - x_t)^2 \tag{21}$$

4.2.2 Relational Dependency Network-based Approach

We use the online-order processing example to illustrate the relevant service dependencies, namely QoS dependencies. In this example, we have 8 services. Service S1 is responsible for products viewing and browsing, while service S2 identifies the costumer location to allow service S3 to check the products availability. Then, service S4 gets the products and the total price is calculated by service S5. Afterwards, the costumer pays through service S6 along with service S7 for currency conversion. Finally, service S8 delivers the products to the costumers.

In this use case, a number of QoS dependencies can exist. Changes to the QoS values of constituent services affect the QoS values of each other (horizontal dependency). Furthermore, the QoS values of the composite service are affected by the QoS of the constituents, and the other way around (vertical dependency). For example, if all or some constituent services are provided by the same provider, data transmission would happen at a faster rate and the response time of the composite service would be reduced. Also, the price of the constituent services could become cheaper since some service providers offer cheaper prices for services that come together. Consequently, the price of the composite service would be cheaper if the constituents are provided by the same provider. Thus, the price of the composite service depends on the prices of the constituents. On the other hand, the price of the constituents is affected by the composite price. For example, to reduce the price of a composite service, the price of it constituents can modified either by price renegotiation or by replacing the constituent with a cheaper substitute. This is an example of cyclic QoS dependency, in particular, vertical cyclic dependency. Furthermore, there is an example of horizontal cyclic dependency because how the costumers and products are located. We found that S8 depends on S6 which in turn depends on S7. S7 depends back on S2 which depends on S5 that relies on S8. This creates a cyclic dependency chain among services (i.e., horizontal cyclic dependency). Our proposed approach for SLA management considers these dependency relations which enhances SLA violation prediction.

Although service dependencies have an impact on the composite service SLA management, there is no direct description for them [91]. Service dependency information is implicitly described in the SLA, and current SLA management approaches are limited with regard to capturing service dependencies. Most approaches assume that the constituent services are independent which is impractical. Therefore, this work proposes a model to learn the service dependency relations. Furthermore, composite services data is relational in nature, yet current approaches rely on propositional data assuming the constituent services

are homogeneous and statistically independent ignoring the relations among those constituents. Since the values of the same variable, e.g., response time, for related constituents are statistically dependent, relational learning and inference techniques are necessary for composite services. Such techniques improve the SLA violation predictions accuracy.

Representation of real-world data as homogeneous, independent and identically distributed (i.i.d.) instances leads to statistical bias in the results. Therefore, this thesis expresses and reasons on service dependencies in a relational setting. Particularly, we view composite service trust as the probability of compliance with SLA rules. To this end, we propose a relational dependency network-driven model to predict SLA violation and estimating the probability of SLA enforcement considering different types of dependencies in a relational setting.

Relational Dependency Network Learning

Figure 18 shows the relational database with one-to-many associations between a composite service and the constituents. We define the properties of objects (i.e. services) and relations among them as a set of predicates $\mathscr{P}$. Then, we define the associated range for each $p \in \mathscr{P}$. Probabilistic predicates represent the random variables in the domain. Finally, we declare a set of random variable declarations RVD that defines the random variables. Accordingly, we define RDN as a tuple $(\mathscr{P}, RVD, dep)$, where dep is a function mapping each $p \in \mathscr{P}$ to a dependency statement [69].

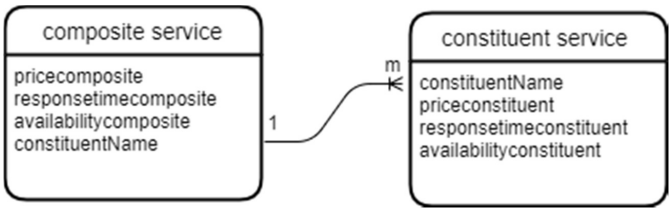


Figure 18: Relational Database

For example, we generate predicates about the objects (i.e. *compositeservice* and

constituent) as follows:

$$priceConstituent/1$$
$$availabilityConstituent/1$$
$$responsetimeConstituent/1$$
$$priceComposite/1$$
$$availabilityComposite/1$$
$$responsetimeComposite/1$$
$$constitutes/2$$
$$collaborates/2$$

with the following random variable declarations:

$$random(priceConstituent(S)) \leftarrow service(S)$$
$$random(availabilityConstituent(S)) \leftarrow service(S)$$
$$random(responsetimeConstituent(S)) \leftarrow service(S)$$
$$random(priceComposite(C)) \leftarrow composite(C)$$
$$random(availabilityComposite(C)) \leftarrow composite(C)$$
$$random(responsetimeComposite(C)) \leftarrow composite(C)$$
$$random(constitutes(C,S)) \leftarrow composite(C), service(S)$$
$$random(collaborates(S,S1)) \leftarrow service(S), service(S1)$$

and ranges:

$$range(priceConstituent(S)) \text{ is } \mathbb{R}$$
$$range(availabilityConstituent(S)) \text{ is } \mathbb{R}$$
$$range(responsetimeConstituent(S)) \text{ is } \mathbb{R}$$
$$range(priceComposite(C)) \text{ is } \mathbb{R}$$
$$range(availabilityComposite(C)) \text{ is } \mathbb{R}$$

$$range(responsetimeComposite(C)) \text{ is } \mathbb{R}$$

$$range(constitutes(C,S)) \text{ is } Boolean$$

$$range(collaborates(S,S1)) \text{ is } Boolean$$

S/$S1$ and C are variables for individual service and composite service respectively. The values of price, response time and availability are numeric and their lower and higher bounds are controlled by the SLA. The value of the predicate $constitute$ is $Boolean$ to indicate whether a service S is a constituent of the composite service C or not, and the $Boolean$ value of the predicate $collaborates$ indicates whether there is a collaboration between services S and $S1$.

Random variable dependency is determined by relational features that represent the relational information of the problem domain. To define the space of relational features, given a set of logic variables L, we enumerate all conjunctions of random variables-value tests U in the form $P = v$ where P is an atom and $v \in range(P)$. Then, we generate a candidate feature $agg_L(A,U)$ where A is an atom and agg is an aggregation function applicable to $range(A)$. There is a number of aggregation functions (functions that map every finite multiset of elements from a domain to a single value from a range R) to be used such as SUM. For example, using the interpretation I that assigns a value to each random variable from its range, we define a relational feature of the form $agg_L(A,U)$ to compute the composite price as follows:

$$service(s1)$$

$$service(s2)$$

$$service(s3)$$

$$service(s4)$$

$$compositeService(c1)$$

$$priceConstituent(s1,30)$$

$$priceConstituent(s2,40)$$

$$priceConstituent(s3, 35)$$

$$priceConstituent(s4, 30)$$

$$constitutes(c1, s1)$$

$$constitutes(c1, s2)$$

$$constitutes(c1, s4)$$

$$agg_C(priceConstituent(S), constitutes(C, S) = true)$$

where the set of variables L is $\{C\}$ ranging over composite services and the conjunction U consists of $constitutes(C, S) = true$ that tests if a service S is a constituent of a composite service C. Accordingly, to compute the price of a particular composite service $c1$ using the aggregation function SUM, we have a feature as follows:

$$SUM(priceConstituent(S), constitutes(C, S) = true) =$$

$$SUM(priceConstituent(s1), priceConstituent(s2), priceConstituent(s4)) =$$

$$SUM(30, 40, 30) = \$100$$

Finally, we learn dependency statements $(P|Parents(P))$, which defines for each random variable the other random variables that it depends on. In addition, we learn the associated conditional probability distributions (CPDs) to model the distribution of the target predicate P on the parent set of relational features $Parents(P)$. To this end, a greedy approach is adopted to select the features in the parent set. Feature selection generally requires repeated CPD estimation while measuring the change in the scoring criterion. Each iteration selects one feature to be added to the parent set until no inclusion improves the scoring criterion, namely the pseudo-loglikelihood. After each feature addition, CPD is learned on the training data and then scoring it on the validation data. The procedure terminates if the score becomes stable with no more improvement. Pseudo-loglikelihood (PLL) estimates the contribution for each relevant random variable conditioning on all other substitution values in the interpretation I is calculated as follows:

$$PLL = \sum_{p \in \mathscr{P}} \sum_{g \in gr(p)} log[P(I(g))|I(Parent(g))]$$

To learn the local CPD of each predicate, we define the CPD model to use both the range of the target predicate P and the parent set $Parent(P)$. We use linear Gaussian distribution since the predicate's range is continuous and all the features in the parent set have continuous values. Finally, the inference in RDN is performed by using an ordered pseudo-Gibbs sampler. Ordered Gibbs sampling randomly initiates each random variable, and then iterates over the variables in a fixed order and resamples the value of each variable from its CPD [36].

In other words, given a set of random variable declarations RVD for all probabilistic predicates in $\mathscr{P}$, we learn the parent set of each random variable *randvar*. Particularly, we learn a local distribution that models each p using the set of candidate features for p. Consequently, conditional probability distribution and a dependency statement will be associated with each *randvar* that maximizes the pseudo-loglikelihood score. The final model is obtained by conjoining all learned local distributions to specify the joint distribution over the random variables. To this end, an ordered Gibbs sampler is applied to CPDs to get the joint distribution.

Accordingly, the trust toward a composite service is constructed by performing N iterations of the ordered pseudo-Gibbs sampling. It is computed by Equation 3 that defined in Subsection 4.1.1.

4.3 Experiments

We implement our experiments using a machine having the following characteristics: CPU: Intel Core i7-4790; Processor: 3:60 GHz; Operating System: 64-bit Windows 7; RAM: 16 GB. The evaluation of our framework is carried out using WS-DREAM dataset [71] that includes real-world QoS measurements (i.e. response time ((sec)) and throughput (kbps))

from 142 service users on 4,500 services over 64 consecutive time slices at a 15-minute interval. The dataset is publicly released for the research in the computational services community to represent the population in a fairway. This dataset has been largely accepted in the community as comprehensive and highly representative, which makes it widely adopted in the relevant literature on services computing.

4.3.1 Direct Trust Prediction Accuracy

In this experiment, we explore the efficiency of the service direct trust prediction. In this experiment, we vary the training data density from 100 to 500 in order to estimate the accuracy of the learned dependency network-based trust model. To this end, the log score [36] is used to measure the prediction accuracy of the model by Equation 22 on the corresponding test set $(x_1,...,x_S)$. This score reports the average of log probability values across the test set. This means that on average, the log probability that each output variable assigns to the given value in the test case given the values of all other input variables (variables that are used only to predict output variables) is as follows:

$$Score(x_1,...,x_S|model) = -\frac{\sum_{i=1}^{S} log_2 p(x_i|model)}{nS} \tag{22}$$

where for each Gibbs sampler invoked to determine $p(x_i|model)$, we average 5000 iteration.

Figure 19 depicts the evolution of the dependency network prediction accuracy where the x axis represents the density of the training data and the y axis indicates the accuracy of the learned model on the test data. Furthermore, we vary the number of input variables that are used to predict the output variables from 10 to 50. As illustrated by the figure, the prediction accuracy is related to the density of the training data. This result indicates that the prediction accuracy of the dependency network increases as more data accumulated in

the model. In other words, the prediction accuracy of the dependency network increases as more information is accumulated for the evidence, i.e. input variables.

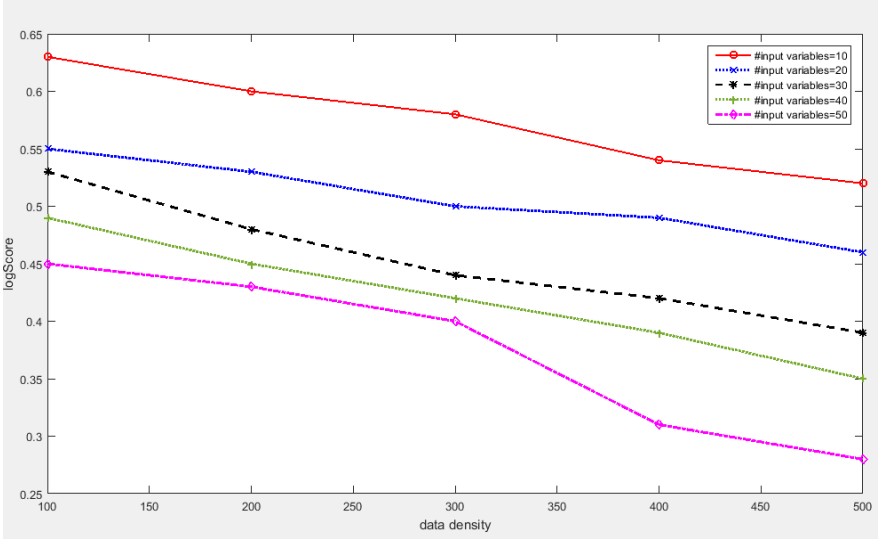
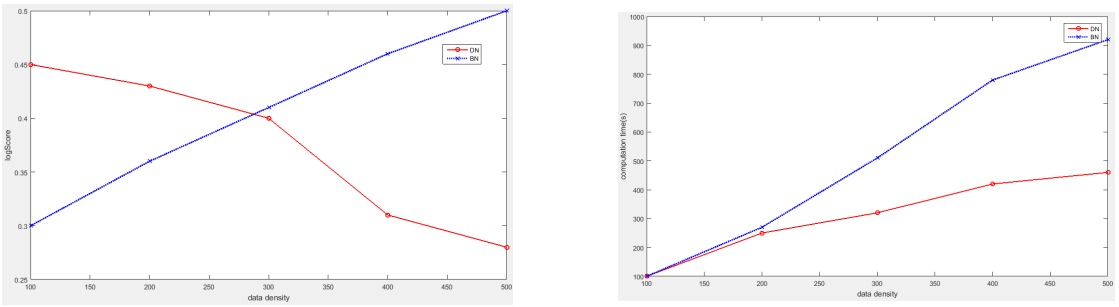


Figure 19: Dependency network performance

4.3.2 Dependency Network vs Bayesian Network

This experiment compares the performance of the dependency network-based trust model (DN) and the Bayesian network-based trust model (BN). The joint probability from a Bayesian network is determined using the law of total probability. We compare the prediction accuracy of the two models. Then, we compare the computational efficiency for learning the models.

Figure 20a depicts the difference of the prediction accuracy of the models, where we fix the number of input variables to 50. As shown in the figure, the Bayesian network-based trust model shows higher prediction accuracy for small training data sets than the one shown by the dependency network-based trust model. This is attributed to the fact that the Bayesian network utilizes an exact inference algorithm which results in better quality compared to the dependency network. However, as more data accumulated into the



(a) (b)

Figure 20: Dependency network-based approach (DN) vs. Bayesian network-based approach (BN). (a) Prediction accuracy of the models. (b) Computational efficiency for learning the models.

models, contrary to the Bayesian network-based model, the prediction accuracy of the dependency network-based model increases. The decrease in the prediction accuracy of the Bayesian network-based model proves the limitation of the Bayesian Network in capturing dependency relations that deteriorate its prediction accuracy. This result implies that the proposed dependency network-based trust model is more suitable for large and complex environments. Moreover, it proves its ability to capture dependency relations, which improves its prediction accuracy.

Afterward, we measure the computational efficiency of the models on a larger scale by considering the computation time for learning the models. Figure 20b plots the relationship between the learning time, **y** axis, of both models and the training data density, **x** axis, where we fix the number of input variables to 50. As shown in the figure, the proposed dependency network-based model is superior to the Bayesian network-based model in terms of learning time. This is because the Bayesian Network-based model uses an exact algorithm which is known to be NP-complete, whereas the proposed dependency network-based model uses an approximation algorithm. This proves the robustness of the proposed approach in large-scale settings.

4.3.3 Subjective Trust in Dynamic Environment

This experiment explores the efficiency of our prediction approach of the service's subjective trust in dynamic environments. We estimate the log score by Equation 22 to assess the model's prediction accuracy. To simulate the concept drift in data, we use a random threshold to define the errors and hits made by the predicted model, such that, if $p(X_i|model)$ for each instance X_i in the windows of the data is less than the threshold, an error is reported for the corresponding window, otherwise it is a hit. This threshold is changed periodically. The concept drift is imitated by increasing the number of errors in the recent window of data.

Figure 21 depicts the evolution of the predictive accuracy of the static model based on traditional dependency network and the adaptive one in dynamic environments, where the x axis represents the dimensionality of the space, i.e., the number of input variables. The secondary vertical axis represents the concept drift level, which is the percentage of test instances that have a change in the relationships between input and output variables. The figure shows that our approach outperforms the traditional one by 20%. This contributes to the ability of the proposed approach to rapidly adapt to the changing environment. In addition, the prediction accuracy is correlated with the dimensionality of the space. In other words, the prediction accuracy of the proposed approach increases with the amount of available relevant information.

4.3.4 Accuracy and Resiliency Numerical Results

This experiment analyzes the accuracy and resiliency properties of the proposed collusion detection approach against collusion attacks. We compare our approach with PeerTrust-PSM proposed in [92], which is based on a personalized similarity measure (PSM) and QoS Correlation Trust introduced in [58], which is based on the personal experiences of the truster.

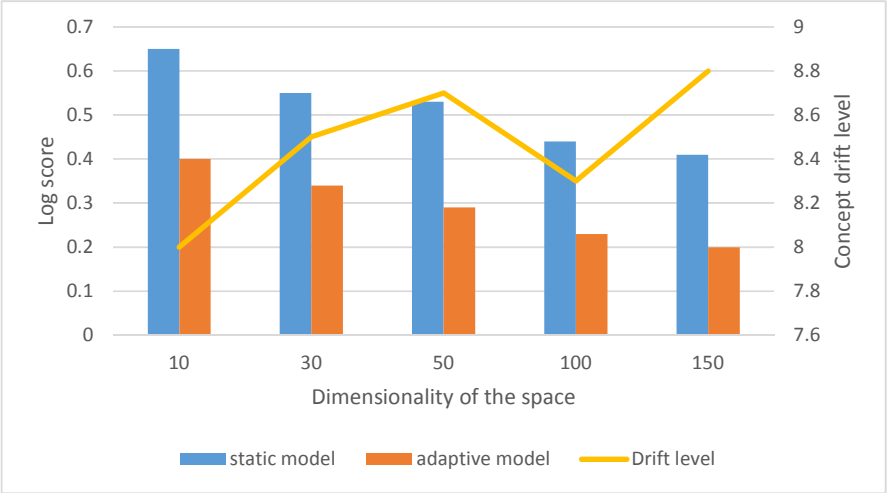


Figure 21: Adaptive dependency network performance

Due to the current limited availability of real-world datasets that report feedback rating data [58], we follow the same simulation setup as in [58]. We set the number of interacting nodes, i.e. services and the corresponding referees, to 100 and simulate service trust in 500-time steps. At each step, a random number of interactions occur with each of the nodes. The initial objective trust value of each node is 0. Each referee gives a service a rate that represents its actual performance on a scale of 10. We use the utility function, i.e. a single metric to quantify the quality perception of the delivered service, proposed in [95] to calculate the service performance, where all QoS metrics are weighted by their importance and normalized by their averages and standard deviations, so they are not biased by any metric with a large value. Finally, the objective trust of the services is updated based on the collected feedback ratings by calculating the mean value. The final objective trust values are the average of 100 runs.

We compare the estimated service objective trust value of our approach against QoS Correlation Trust and PeerTrust-PSM approaches by reporting the estimation error of the objective trust of one service as indicated by a referee and the one calculated during the multi-round simulation. Figure 22 plots the mean errors of the aggregated trust based

on feedback from 99 referees. As it is illustrated by the figure, we vary the percentage of colluders between 10% and 90% and set the percentage of malicious feedback from them to 100%. The results show that the proposed approach performs better than other approaches. This out-performance is justified by the fact that our approach relies on global quality metrics based on nodes' history of interactions. However, our approach is limited to detect the colluders of type A, since they are independently malicious and do not form a collectively malicious group.

Also, Figures 23 and 24 show the objective trust distributions of the nodes without and with the proposed collusion detection approach respectively. We set the percentage of colluders to 10% and the percentage of malicious feedback to 100%. We consider three types of colluders: B, C, and D. Type B and D colluders boost the trust of the nodes of type B whereas type B and C colluders lessen the trust of the good nodes. Good nodes are those which provide good services while having good behaviour. We consider the trustee as a good node in this experiment. For clarity, we plot only the trust values of the colluders and the trustee. As the figures show, the objective trust of the trustee, represented by the left bar, is increased by our approach. This is due to the fact that colluders receive 0 trust values after their detection and they are excluded as referees. In addition, Figure 24 shows the ability of our approach to detect all different types of colluders. Therefore, detecting and excluding the colluders improve the effectiveness of service objective trust evaluation.

4.3.5 Bootstrapping Trust Prediction Accuracy

This experiment explores the effectiveness of the proposed trust bootstrapping mechanism in providing accurate initial trust values. The bootstrapper initiates a sample training dataset from its historical records of similar services to the newcomer based on the functional and non-functional specifications under different contextual environments. After each interaction with the new service, the transaction record is added to the training

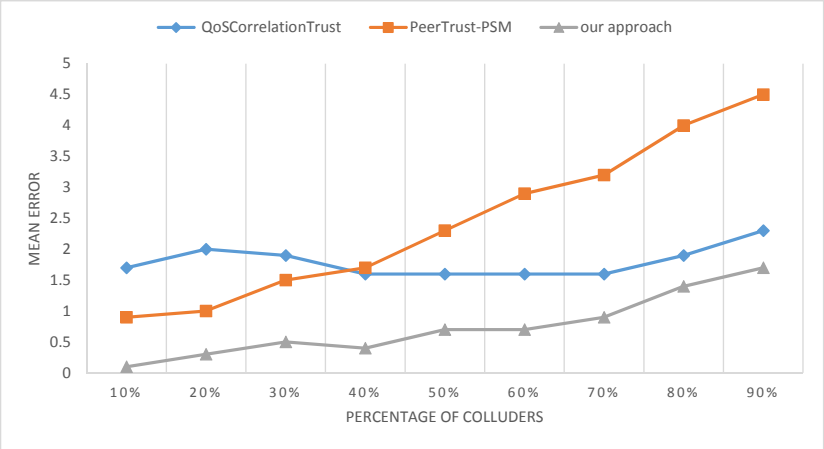


Figure 22: Service objective trust evaluation under collusion attacks

dataset. Hence the ensemble classifier is updated towards more accurate prediction.

To the best of our knowledge, there is a lack of real datasets that map the QoS values to a behaviour class label. Hence, we modify the WS-DREAM dataset by adding behaviour class labels $\{trust, malicious\}$ for each service. We use the utility function proposed in [95] to have a single quality metric whose value ranges between 0 and 1 for each service. The metrics above 0.75 will be assigned class label *trust*, otherwise class label *malicious* will be assigned. For the KNN algorithm, we set K to the square root of the number of training samples.

Figures 25 and 26 plot the ROC (Receiver Operating Characteristic) curves generated by our classifier. The two metrics considered in these curves are sensitivity and specificity. Sensitivity measures the percentage of positives that are correctly classified as such (i.e., true positives). Specificity, on the other hand, measures the percentage of negatives that are correctly classified as such (i.e., true negatives). Thus, 1− specificity indicates the percentage of negatives that are wrongly classified as positives (i.e.,

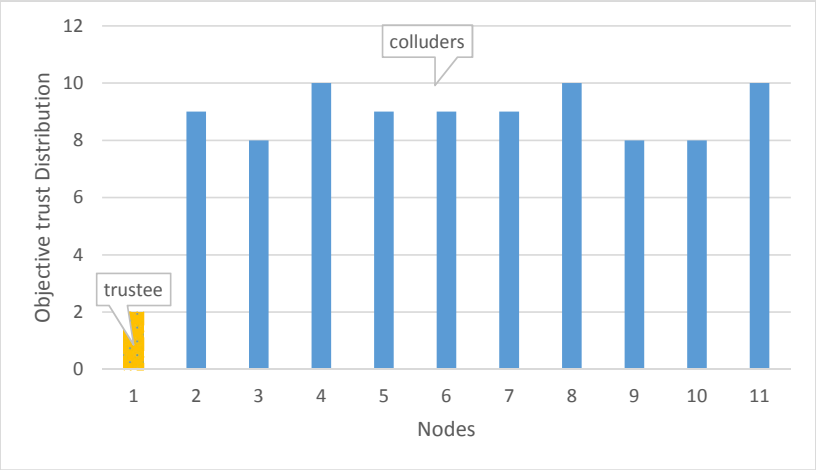


Figure 23: Service objective trust distributions without our approach

false positives). In other words, $1-$ Specificity $= \frac{FalsePositives}{FalsePositives+TrueNegatives}$, and Sensitivity $= \frac{TruePositives}{TruePositives+FalseNegatives}$. Each point on the ROC curve represents a sensitivity/(1-specificity) pair where the point which coordinates are $(0,1)$ represents 100 percent sensitivity and 100 percent specificity. Therefore, a classification model with perfect discrimination has a ROC curve that passes through this point. Thus, the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the model. Moreover, the overall classifier's accuracy is quantified in terms of Area Under the Curve (AUC), where the bigger the area covered, the better the classification model is at distinguishing the given classes. The ideal value for AUC is 1 whereas a value of 0.5 is worthless.

Since the used ensemble classifier yields a probability for the class label of the observation, we use a threshold to produce a discrete classifier. Each different threshold value produces a different point in ROC space (corresponding to a different confusion matrix). For detecting malicious services, Figure 25 depicts the prediction accuracy of the proposed classification-based bootstrapping solution. It is worth mentioning that the words positive and negative refer to the presence or absence of the condition, respectively. Thus, in this figure, sensitivity refers to the percentage of malicious services that are correctly classified as such and $1-$specificity refers to the percentage of malicious services that are classified as

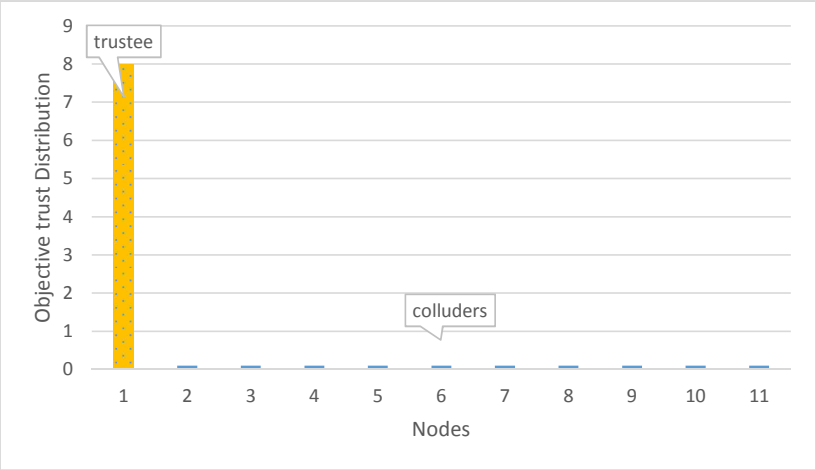


Figure 24: Service objective trust distributions with our approach

trustworthy. However, since Figure 26 measures the accuracy of the bootstrapping mechanism in classifying the trustworthy services as such, sensitivity refers to the percentage of trustworthy services correctly classified as such, whereas 1−specificity means the percentage of trustworthy services that are classified as malicious. As it is shown in the figures, our bootstrapping approach produces high AUC values reaching 0.979, which proves its efficiency.

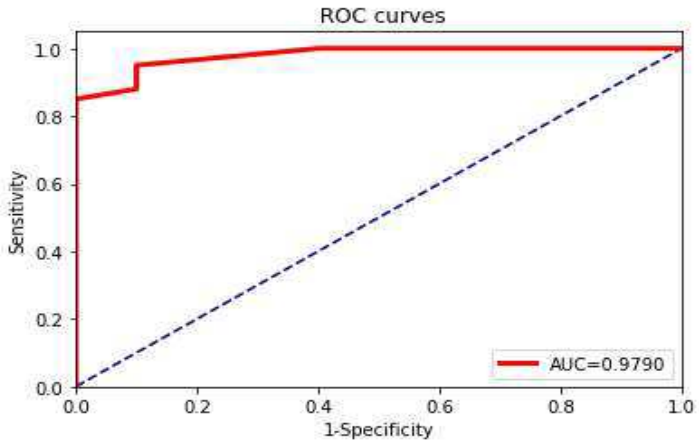


Figure 25: Accuracy in classifying malicious services

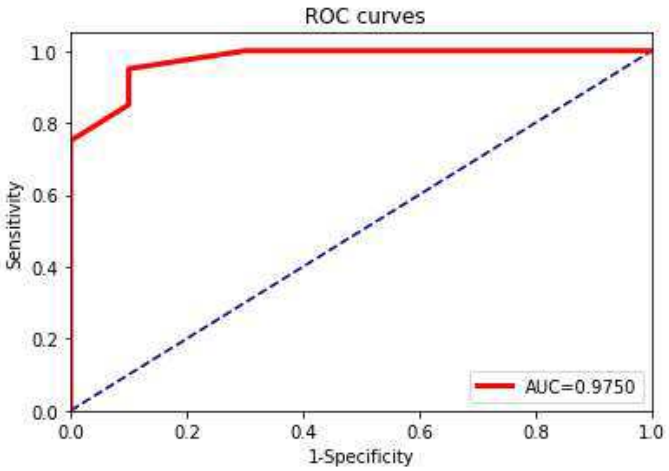


Figure 26: Accuracy in classifying trustworthy services

4.3.6 Kalman filter based SLA Prediction Accuracy

This experiment evaluates the accuracy of the proposed online QoS prediction approach against other prediction approaches [17] and [99].

Average: The average QoS value at each time window is used as the predicted value, this method is used in [17].

MF: In [99], the authors applied matrix factorization [70] to the QoS prediction.

Since the Kalman filter algorithm requires a small amount of data to build the model, the density of the dataset varied from 5% to 30% by removing QoS records from the dataset. Those QoS records are used as training data, while the removed records are used for testing and evaluating the prediction accuracy. We use the median relative error (MRE) metric to evaluate the prediction accuracy of the proposed approach against the other approaches. MRE is the median value of all the pairwise relative errors:

$$MRE = median_{k=0,t}\{|\hat{x}_k - x_k|/x_k\}, \tag{23}$$

where $\hat{x}_k$ is the predicted value corresponding to the real value x_k, and t is the number of time windows. We set the time window length to 1 minute for 20 time windows. Figure 27 shows MRE for response time and Figure 28 shows MRE for availability where y-axis indicates MRE and x-axis refers to data density. We notice from the figures that the accuracy of the proposed approach is independent of the data density, unlike other approaches. This is possible because our approach does not require historical data, proving the efficiency of our approach.

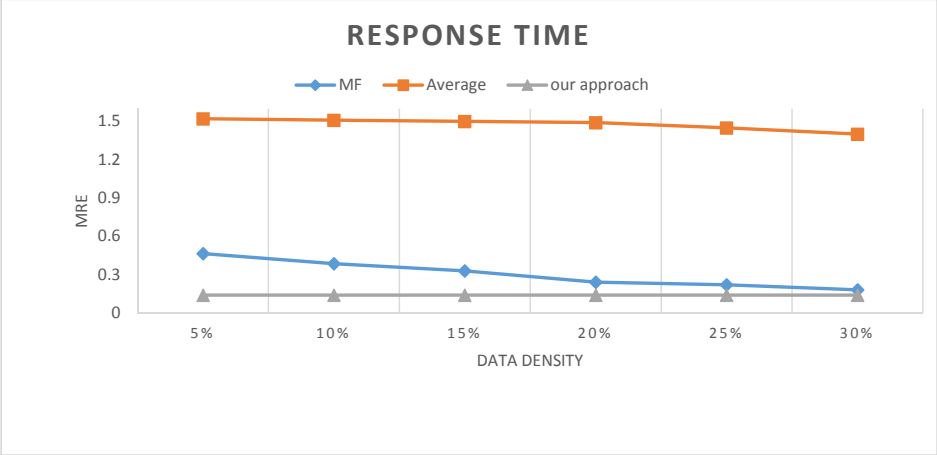


Figure 27: MRE for response time

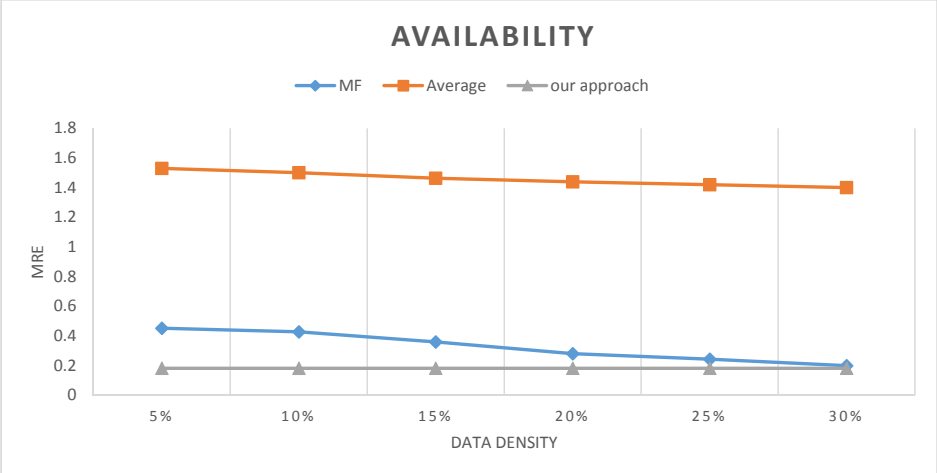


Figure 28: MRE for availability

4.3.7 Relational Dependency Network-based SLA Prediction Accuracy

This experiment is based on the implementation published in [69] where the authors used a Java program to perform the learning and a Prolog program to compute the value of a feature. We compute the QoS of composite services using the aggregate functions defined in [34]. We generate a handcrafted model to compare the learned model against it. We define, through a set of dependency statements, different dependency relations that could exist among composite service and the constituents based on the state-of-the-art. The handcrafted model is defined as follows:

$$priceConstituent(S)|value_{\{S\}}(priceCompsite(C), constitutes(C,S) = true)$$

$$priceConstituent(S)|value_{\{S\}}(priceConstituent(S1), collaborates(S,S1) = true)$$

$$availabilityConstituent(S)|value_{\{S\}}(availabilityCompsite(C), constitutes(C,S) = true)$$

$$availabilityConstituent(S)|value_{\{S\}}(availabilityConstituent(S1), collaborates(S,S1) = true)$$

$$responsetimeConstituent(S)|value_{\{S\}}(responsetimeCompsite(C), constitutes(C,S) = true)$$

$$responsetimeConstituent(S)|value_{\{S\}}(responsetimeConstituent(S1), collaborates(S,S1) = true)$$

$$priceComposite(C)|sum_{\{C\}}(priceConstituent(S), constitutes(C,S) = true)$$

$$availabilityComposite(C)|product_{\{C\}}(availabilityConstituent(S), constitutes(C,S) = true)$$

$$responsetimeComposite(C)|sum_{\{C\}}(availabilityConstituent(S), constitutes(C,S) = true)$$

Figure 29 plots the quality of the probability estimation of the learned model using the weighted pseudo-loglikelihood (WPLL) metric that is the sum of PLLs for each predicate divided by the number of groundings of that predicate in the interpretation. We vary the

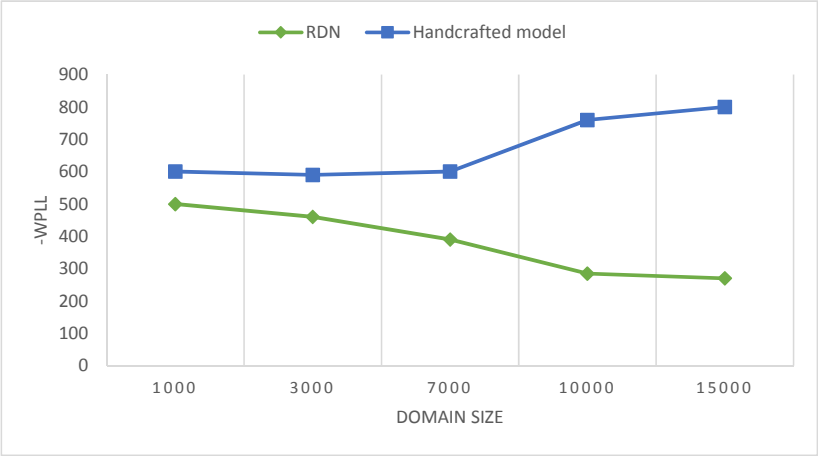


Figure 29: RDN-based model vs handcrafted model

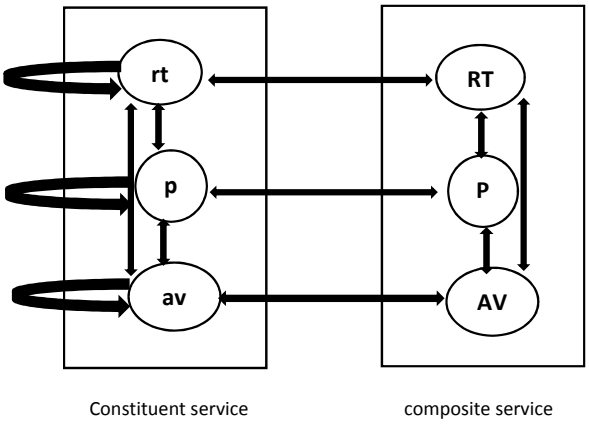


Figure 30: Graphical representation of RDN based trust model for a composite service; rt/RT,p/P, av/AV represent response time, price and availability of constituent/composite service

domain size (i.e., the number of the constituents) from 1000 to 15000. The figure shows that the learned model outperforms the handcrafted one. This is attributed to the fact that the proposed model can capture more relations that are not considered by the handcrafted model. We found a considerable number of bi-directional dependencies among the relevant random variables. Figure 30 depicts an RDN for a composite service. As it is shown, the price of the composite service depends on the price of the constituents and vice versa, and the price of the constituent service depends on the price of other constituents. Besides, QoS values of a service (constituent or composite) depend on each other. This is because the providers deliver high QoS values for high price and degrade the delivered QoS for low prices, and users pay for the received QoS values of the service [96]. In other words, QoS degradation may change the behavior of the user to pay less which in return leads the provider to degrade the delivered QoS.

4.4 Conclusion

The proposed framework for managing context-aware trustworthy services provides comprehensive trust management including subjective, objective, and bootstrapping trust. We capitalized on the dependency network to estimate the probability of providing an acceptable level of QoS (subjective trust). We modified the traditional dependency network to consider the dynamic cyclic dependency relations that relate QoS metrics to context variables. We used the statistical log score to assess the model's prediction accuracy. The results revealed that the proposed approach outperforms the traditional one in dynamic settings. Furthermore, we proposed a service objective trust evaluation technique that enabled us to detect collusion attacks. Unlike the existing related work, we introduced a direct collusion detection method that exposes colluding attackers who provide fake or misleading trust feedback. We employed the estimation error of the objective trust as indicated by referees relative to the one calculated by the multi-round simulation. Our experiments

showed that detecting and excluding the colluders directly improves the effectiveness of service objective trust evaluation. Furthermore, we proposed a trust bootstrapping mechanism that capitalized on the service observations. Particularly, we adopted an ensemble of classifiers to assign newcomer-services initial trust values. The ROC (Receiver Operating Characteristic) curves are used to measure the accuracy of the classifier used in the proposed trust bootstrapping mechanism. The efficiency of the proposed mechanism is proved by high AUC values.

The main findings of this chapter are about the new intelligent trust model of autonomous services that considers their dynamically changing environments. The first finding is that the prediction of the provided QoS shows better results when QoS is dynamic and responds to context environment changes by leveraging the dynamic dependency network linking the QoS metrics and context variables of the environment. The second finding is that the objective trust performs better when it is resilient to collusion attacks by leveraging the power of mass collaboration among referees. The third finding is that the bootstrapping mechanism that observes the behaviours of new comer services with no trust resources using the concept of social adoption to estimate their initial trust values excels by being resilient to white-washing attacks. Thus, the major practical implication of the present research is to provide comprehensive trust management that enables service providers to (1) maintain user satisfaction; (2) secure provided QoS; (3) maintain service reputation; (4) involve newborn services in transactions to increase the profit; and (5) avoid high economic compensation caused by SLA breach penalties. Providing secure, trustful and reputable platforms for computational services benefits a large spectrum of businesses, particularly in the modern era of data and artificial intelligence-driven applications.

Finally, we proposed an on-line context-aware QoS prediction approach based on the Kalman filter algorithm that enhances SLA violation prediction accuracy. However, this model does not consider service dependency that exists due to service collaboration towards

the composition goals. Therefore, we proposed a trust-based model for SLA management of composite services based on relational dependency network (RDN). We leveraged horizontal, vertical, and cyclic QoS dependency relations in a relational setting, that have been neglected in the literature. Experimental results show that our proposed model achieves higher accuracy compared with a handcrafted model that captures QoS dependencies that are mostly known in the literature.

In summary, this chapter has achieved the second and the third research objectives (Objectives 2 and 3) discussed in Chapter 1, which aimed at predicting SLA and/or QoS violation of the composite service and the constituents to avoid malicious performance. The proposed prediction models allow to take the appropriate adaptation actions that will be discussed in the next chapter.

Chapter 5

Service Adaptation Actions

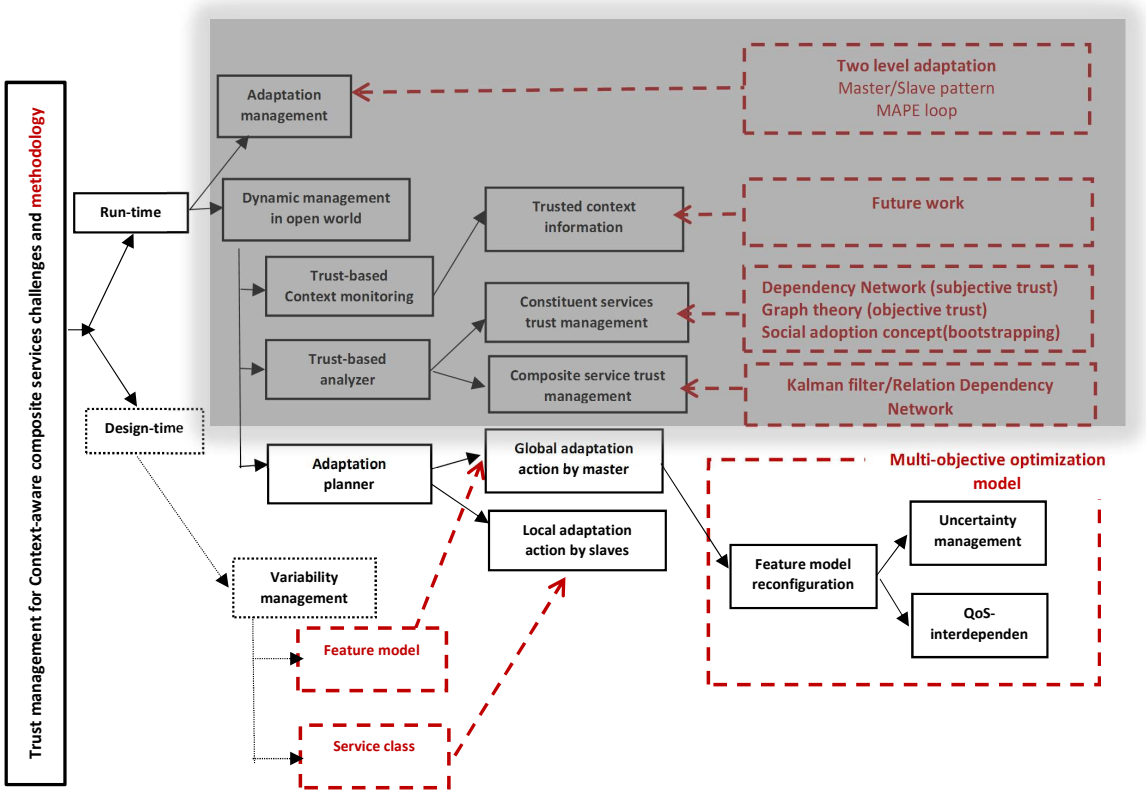


Figure 31: Chapter 5 challenges

Figure 31 (unshaded part of the figure) shows the addressed challenges in this chapter. Consequently, this chapter discusses the adaptation actions that are triggered to prevent QoS degradation of the running services upon QoS/SLA violation prediction. In Section 5.1, we present the proposed adaptation approach that enables the master, particularly in the plan step of the MAPE loop, to reconfigure the composite service using the feature model. In Section 5.2, we discuss the proposed local adaptation action that enables slaves to replace failed constituent services promptly, which reduces the time complexity of reconfiguring the composite service.

5.1 Composite Service Adaptation

A composite service can have alternative variants that provide different QoS values, therefore, it can add new services or discard others in response to encountered changes. In order to manage runtime variability, we apply ideas from software engineering, particularly dynamic software product line (DSPL) [32]. The dynamic variability management of DSPL is handled by a feature model [44]. The feature model is configured by activating/deactivating feature nodes to adapt the workflow of the composite service. Each configuration has to satisfy the constraints of the feature model, which is complex due to the numerous number of possible configurations even in a small feature model. Furthermore, different features have different impacts on performance and expose different QoS values. To tackle this challenge, we model the feature model configuration as an optimization problem that searches for the optimal feature configuration.

A composite service uses a combination of resources such as CPU and memory to satisfy its SLA constraints, which will represent the cost of this composite service. However, cost constraints can be violated during the adaptation process. Therefore, adaptation actions should comply with the cost constraints to avoid situations in which the cost of resource consumption exceeds the capacity cost. To address this challenge, we include cost

constraints along with feature model constraints in our optimization problem while considering minimizing the total cost in terms of resource consumption and monetary violation costs. Since this objective conflicts with the objective of performance maximization described in the previous paragraph, we model the problem as a multi-objective optimization and propose a genetic-based algorithm to solve it.

Therefore, an effective composite service adaptation action requires selecting a set of feature nodes that optimally satisfies the global QoS constraints while minimizing the cost. We consider the availability metric QoS^*_{avail} as an example of positive QoS constraint, i.e., a higher value indicates a better QoS, while the response time metric $QoS^*_{resp\_time}$ and the cost QoS^*_{cost} as negative constraints, i.e., higher values indicate worse QoS. In addition, we deliberate the interdependency relations among QoS constraints to address and resolve conflicts. We model the composite service adaptation problem as a multi-objective optimization problem that optimizes feature nodes selection.

Before formalizing the optimization problem, we first define the key concepts used in the composite service adaptation.

Definition: Resource (i.e., CPU or memory) consumption rc^j of a feature f_j is defined as the cost of resource usage r^j_{usage} during a period of time: $rc^j = r^j_{usage} \times r^h_{price} \times T$ where T is the number of usage hours, and r^h_{price} is the pricing fee per hour, which is assumed to be static in this thesis.

As example of pricing fees, we use a snapshot of Amazon pricing schema [4]: \$0.018 per GHz/h for CPU and \$0.025 per GB/h for memory.

Definition: Cost of a feature f_j is the total resource consumption. Let $A = \{cpu, mem\}$, we define cost as $qos^j_{cost} = \sum_{a \in A} rc^j_a$.

Definition: Global SLA is defined as a 3-tuple $SLA = < QoS^*_{cost}, QoS^*_{resp\_time}, QoS^*_{avail} >$, where $QoS^*_{resp_time}$ represents the lower bound constraint on response time, QoS^*_{avail} is the upper bound constraint on availability, and QoS^*_{cost} is the cost constraint.

Definition: A feature model is defined as a set of features $F = \{f_1, f_2, \ldots, f_m\}$, where m refers to the total number of features.

Definition: Each feature $f_j \in F$ is expressed as 3-tuple $f_j = < qos^j_{cost}, qos^j_{resp\_time}, qos^j_{avail} >$.

Let S be the set of predicted values of QoS metrics estimated by Kalman-based prediction model, $S = \{s_{cost}, s_{resp\_time}, s_{avail}\}$, When a metric falls behind the SLA constraint, a penalty will be applied for each violation. Penalty is defined in the specifications of the global SLA agreements and expressed as a function $P(s) : R \rightarrow R$. The penalty values increase monotonically, i.e., higher number of violations results in a higher total penalty value.

Accordingly, we calculate the total penalty value for each violated value $s \in S$ as follows:

$$P = \sum_{s \in S} P(s), \qquad if \ \ s \ \ is \ \ violated \tag{24}$$

Given a running composite service, a global SLA, a feature model, and the feature constraints $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, adapting this composite service is a process of activating and deactivating features form the feature model F that optimally satisfy the global SLA constraints and feature constraints C.

The response time and availability metrics of the composite service are optimized by delivering QoS values close to the QoS bounds identified in the global SLA. Accomplishing these objectives while minimizing the total cost form the fundamental objectives of the composite service adaptation action. In other words, the optimization problem aims to maximize the performance (availability and response time) of the composite service while minimizing the cost subject to the constraints.

The optimization problem faces the following challenges:

- The adaptation cost minimization.
- Feature model configuration.

- QoS interdependence.

5.1.1 Multi-Objective Composite Service Adaptation

The global SLA constraints have to be satisfied during the optimization process. To achieve this, our approach selects feature nodes that yield aggregate QoS values close to the global SLA constraints by minimizing the Euclidean distance. It is worth mentioning that since QoS values are represented as one-dimensional vectors in our problem, the Euclidean distance is more appropriate than the cosine similarity for the minimization problem. In fact, applying the cosine similarity in such a case will always result in 1, which misleads our selection approach. Moreover, unlike cosine similarity which does not consider the magnitude of the vectors in the calculation, the Euclidean distance supports our objective to minimize the magnitude of the one-dimensional vectors to optimize QoS performance by selecting the closest features that satisfy the SLA constraints. The approach is as follows:

- Minimize the distance for the availability property between the global SLA constraint (QoS^*_{avail}) and the aggregate availability value of the selected feature nodes as expressed in Equation 25.

- Minimize the distance for the response time property between the global SLA constraint ($QoS^*_{resp\_time}$), and the aggregate response time value of the selected feature nodes as expressed in Equation 26.

- Minimize the total cost and penalties as expressed in Equation 27.

Let fs_j be a boolean variable s.t. $fs_j = 1$ if the feature f_j is selected and $fs_j = 0$ otherwise. Also, let $fs \subseteq F$ s.t. $\forall f_j \in fs$, $fs_j = 1$. Thus, the multi-objective optimization that computes the set of selected features fs can be formalized as follows:

$$min \sqrt{|QoS^*_{avail} - (\prod_{j=1}^{m} qos^j_{avail} * fs_j)|^2} \tag{25}$$

$$min \sqrt{|(\sum_{j=1}^{m} qos^{j}_{resp\_time} * fs_j) - QoS^{*}_{resp\_time}|^2} \tag{26}$$

$$min \quad P + \sum_{j=1}^{m} fs_j * qos^{j}_{cost} \tag{27}$$

where:

$$\sum_{j=1}^{m} fs_j * qos^{j}_{cost} \leq QoS^{*}_{cost} \tag{28}$$

$$fs_j \in \{0,1\} \qquad j = 1,\ldots,m \tag{29}$$

$$fs \quad \text{conforms to} \quad C \tag{30}$$

Theorem 1: The multi-objective composite service adaptation problem (CSAP) is NP-hard.

Proof The multi-objective multidimensional knapsack problem (MOMKP) has been proven to be NP-Hard [54]. The problem of multi-objective composite service adaptation is reduced by MOMKP. We verify the proof by finding a solution that fits for both CSAP and MOMKP. The formation of MOMKP is as follows: given E items I_e $[e = 1,\therefore,E]$ with C characteristics $w^e_c[c = 1,\therefore,C]$, and P profits $r^e_p[p = 1,\therefore,P]$, select the items that maximize the total profit while not exceeding the C knapsack characteristics W_c. CSAP is formulated as a set of m features $F = \{f_1, f_2,\ldots, f_m\}$, corresponding to E items of MOMKP. Each feature $f_j =< qos^{j}_{cost}, qos^{j}_{resp\_time}, qos^{j}_{avail} >$ is annotated by its cost and performance (i.e. response time and availability). For each feature f_j, set response time and availability as the profits of the item I_e, while the cost will be considered as a profit and a characteristic. The content of the knapsack is the feature set that is selected to minimize the distance between the profit and the global SLA constraints while not exceeding the cost. The minimization

objective in the case of CSAP is equivalent to the maximization for MOMKP. Therefore, a solution to CSAP would yield a solution to MOMKP. Since the transformation from MOMKP to CSAP is polynomial, we conclude that the multi-objective composite service adaptation problem is NP-hard.

5.1.2 NSGA-II-based Decision Algorithm

Computational intelligence techniques that include heuristic and meta-heuristic algorithms have demonstrated their ability to solve complex problems (usually NP-hard problems) in many areas [41]. Hence, these techniques can be used to solve composite service adaptation, which is an NP-hard problem. Meta-heuristic algorithms include particle swarm optimization, ant colony optimization, and evolutionary algorithms such as genetic algorithms (GAs). GAs are widely used to solve complex service composition optimization problems as reported in the survey paper published in [41]. In this thesis, we particularly selected the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [27] that has shown better performance over other candidates (such as SPEA2) for multi-objective optimization problems for service composition [52].

We propose an NSGA-II-based decision algorithm to the problem of composite service adaptation as illustrated in Algorithm 2. It starts with an initial population of randomly generated solutions (Line 5). Each solution represents a feature selection in the feature model. In line 6, the algorithm validates the feature selection according to the set of constraints C as shown in Algorithm 3. Then, the set of solutions are evaluated in Algorithm 4 using the previously discussed Equations 25, 26 and 27 (line 7). The population is sorted based on the rank and crowding distance (lines 8, 9) and the fittest solutions are selected to go through a process of evolution based on crossover and mutation operators (lines 11-13). This process creates a child population, which is validated and evaluated similarly to its parent (lines 14, 15). Both populations, the parent and child, are merged and sorted (lines

Algorithm 2 NSGA-II-based decision algorithm

1: **Input:** a feature model $F = \{f_1, f_2, \ldots, f_m\}$ with $f_j = < qos^j_{cost}, qos^j_{resp\_time}, qos^j_{avail} >$,C, global SLA and population size M
2: **Output:** The non-dominated solutions in the final population

3: **Begin**
4: $t \leftarrow 0$
5: $P_t \leftarrow$ RANDOM-POPULATION(M)
6: validate-pop (P_t) //Algorithm 2
7: evaluate-pop (P_t) //Algorithm 3
8: fast-nondominated-sort(P_t)
9: crowding-distance-assignment(P_t)
10: **while** the termination criterion is not met **do**
11: $Q_t \leftarrow$ select (P_t)
12: crossover (Q_t)
13: mutate (Q_t)
14: validate-pop (Q_t) //Algorithm 2
15: evaluate-pop (Q_t) //Algorithm 3
16: $R_t \leftarrow P_t \cup Q_t$
17: fast-nondominated-sort(R_t)
18: crowding-distance-assignment(R_t)
19: $P_{t+1} \leftarrow \emptyset$
20: $P_{t+1} \leftarrow R_t[0 : M-1]$
21: $t \leftarrow t+1$
22: **end while**
23: **End**

16, 25). A new population is filled with the fittest M solutions (line 26). This process continues until a stopping criterion is met. This criterion can be the number of iterations, time, or any other relevant conditions.

The resulting solutions set is located at the first level front, which is known as the Pareto optimal solutions set. This set includes all solutions which cannot be dominated by others. Accordingly, the Pareto optimal solutions set is the set of solutions s s.t. $s_{rank} = 1$.

Algorithm 3 validate-pop (P_t)

```
1: Begin
2:     for each chromosome chro_c ∈ P_t do
3:         for each feature f_j ∈ chro_c do
4:             if f_j ∈ C_1 then
5:                 for each f' ∈ f_j-children do
6:                     chro_c ← chro_c ∪ f^
7:                 end for
8:             if f ∈ C_4 then
9:                 for each f' ∈ f_j-sibling do
10:                    chro_c ← chro_c − {f'}
11:                end for
12:            if f_j includes f' then
13:                chro_c ← chro_c ∪ f^
14:            if f_j excludes f^ then
15:                chro_c ← chro_c − {f'}
16:        end for
17:    end for
18: End
```

Algorithm 4 evaluate-pop (P_t)

```
1: Begin
2:     for each chromosome chro_c ∈ P_t do
3:         Evaluate objective functions using Eq. (25),(26),(27)
4:     end for
5: End
```

5.2 Service Adaptation

Local service adaptation action is triggered to: (1) substitute the failed service constituent when local SLA is predicted to be violated, or (2) select the implementing constituents after composite service reconfiguration. Particularly, the main task of the adaptation slaves is to substitute/select failed/new service constituents.

Each node of the feature model is realized by a service class providing its functionality but with different QoS. The QoS metrics annotated in feature nodes express local SLAs for the slaves. Using local SLA, each slave v ranks its associated service class SC_s based on the utility value, i.e., the aggregate QoS value, of the members and QoS similarity with

the feature node f_j. At run-time, the slave selects the top constituent S_{is} to implement the functionality of f_j, while the other service services will be backed up in case the running service S_{is} fails.

Definition: service class $SC_s = \{S_1, \ldots, S_p\}$ is a collection of p service with the same functionality but different QoS metrics.

Definition: constituent service S_{is} is described by a 3-tuple $S_{is} = < qos^i_{cost}, qos^i_{resp\_time}, qos^i_{avail} >$ identifying cost, response time and availability metrics of service S_{is} respectively.

Computing the utility value of a service S_{is} requires normalizing its QoS values as follows:

$$qos^i_{cost} = \frac{max^s_{cost} - qos^i_{cost}}{max^s_{cost} - min^s_{cost}} \tag{31}$$

$$qos^i_{resp\_time} = \frac{max^s_{resp\_time} - qos^i_{resp\_time}}{max^s_{resp\_time} - min^s_{resp\_time}} \tag{32}$$

$$qos^i_{avail} = \frac{qos^i_{avail} - max^s_{avail}}{max^s_{avail} - min^s_{avail}} \tag{33}$$

where $max^s_o = max_{\forall S_{is} \in SC_s} qos^i_o$, $min^s_o = min_{\forall S_{is} \in SC_s} qos^i_o$, s.t. $o \in \{cost, resp\_time, avail\}$. Accordingly, the utility function of service S_{is} is computed as:

$$U(S_{is}) = qos^i_{cost} + qos^i_{resp\_time} + qos^i_{avail} \tag{34}$$

QoS similarity between service S_{is} and feature node f_j is computed using the Euclidean distance as follows:

$$d(S_{is}, f_j) = \sqrt{cost^2 + resp\_time^2 + avail^2} \tag{35}$$

where,

$$cost = \begin{cases} qos^{i}_{cost} - qos^{j}_{cost}, if qos^{i}_{cost} > qos^{j}_{cost} \\ 0, otherwise \end{cases}$$

$$resp\_time = \begin{cases} qos^{i}_{resp\_time} - qos^{j}_{resp\_time}, if qos^{i}_{resp\_time} > qos^{j}_{resp\_time} \\ 0, otherwise \end{cases}$$

$$avail = \begin{cases} qos^{j}_{avail} - qos^{i}_{avail}, if qos^{i}_{avail} < qos^{j}_{avail} \\ 0, otherwise \end{cases}$$

Finally, service class SC_s is ranked in descending order based on service scores measured by Equation 36. The top service will be selected to implement the functionality of f_j, whereas the remainder service will act as backups.

$$score(S_{is}, f_j) = \frac{U(S_{is})}{d(S_{is}, f_j)} \tag{36}$$

5.3 Experimentation and Results

We conduct a set of experiments to validate the effectiveness of our approach against a set of benchmark measurements using the same setup described in Section 3.3. The NSGA-II-based solver algorithm is implemented in C based on the code written by the author of [27].

5.3.1 QoS Requirements Flexibility

This experiment explores the flexibility of our approach to the user's QoS requirements changes. We compare the ability to react to QoS requirements changes of the proposed approach using feature model with the approach of [17] which counts on the goal model

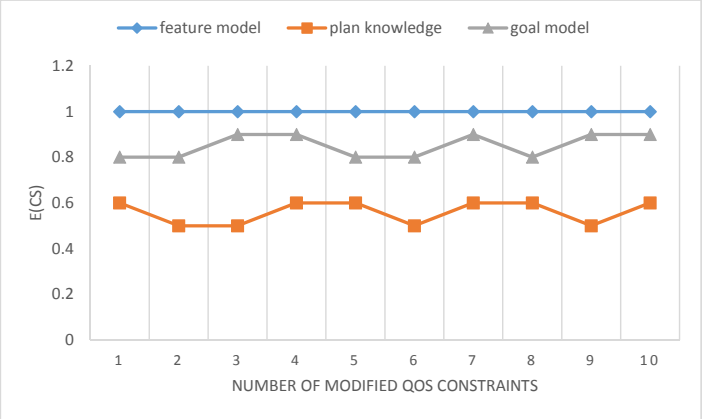


Figure 32: The flexibility of changing user's requirements

and the approach of [7] which relies on knowledge model of different composition plans. For this propose, we measure the efficiency of the composite service $E(CS)$ in terms of the number of the satisfied global *SLA* constraints by Equation 37:

$$E(CS) = \frac{\sum_{QoS^* \in SLA} SAT(QoS^*)}{N} \tag{37}$$

where $SAT(QoS^*)$ returns 1 if the constraint QoS^* is satisfied and 0 otherwise, and N is the number of QoS^* constraints

Figure 32 illustrates the flexibility of QoS constraints modification by the proposed approach and the approaches of [17, 7]. The y-axis denotes the efficiency of the composite service, while we vary the scale of change represented by the number of modified QoS constraints from 1 to 10. The scale of change is denoted by the x-axis. The results show that our approach outperforms other approaches facing QoS constraints modification, those results are justified by the fact that using the feature model enables our approach to capture the variabilities of the composite service. Hence, optimal feature model reconfiguration responds to different user's QoS requirements.

5.3.2 Dynamic Adaptation

In this experiment, we study the efficiency of the proposed adaptive approach towards the changes in dynamic environments. We observe how changes in QoS values affect the execution time. In this experiment, we consider the scale of change represented by the number of service that experience changes in their QoS, and the frequency of change in the QoS values represented by the rate of context-awareness.

We vary the scale of change in the QoS values from 10% to 50%. Figure 33 shows the outcome of the scale of change on the execution time of the proposed approach. The execution time increases from 30*ms* to 170*ms* with the periodic scale of change from 10% to 30% which proves the efficiency of the proposed approach in dynamic environments. The efficiency stems from local adaptation actions that promptly substitute predicted failed service to avoid global violations and composite service reconfigurations. The observed increase in execution time within the scale of change 50% is acceptable since we have a highly dynamic and a fairly complex environment.

To assess the impact of frequency of change on the execution time, we fix the scale of change in the QoS values to 30% and increase the frequency from 10% to 50%. As shown in Figure 34, the execution time dropped from 150*ms* to 28*ms* which underscores the importance of online QoS prediction to increase context-awareness for efficient adaptation.

5.3.3 Local Adaptation

This experiment examines the impact of local service adaptation action on the performance of the adaptation process. We compare the adaptation flexibility and the ability to react to context changes by the proposed approach with the centralized approach [17]. The adaptation actions are recorded during one minute using the initial configuration, config2 [17]. As discussed earlier, a local service adaptation action can be used for (1) service

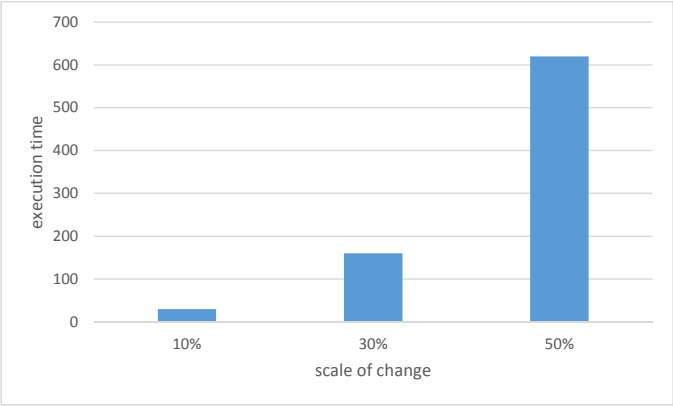


Figure 33: The scale of change

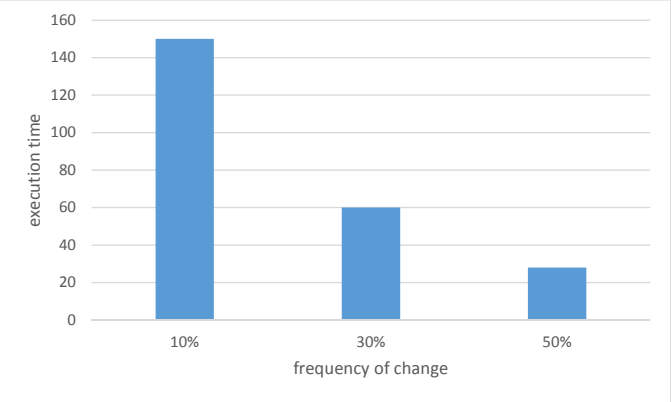


Figure 34: Awareness rate of changing QoS values

substitution, (2) service selection within composite service adaptation action. Figure 35 and 36 illustrate the adaptation flexibility of the centralized approach and the proposed approach, respectively. The x-axis denotes the discrete time intervals of one minute and the adaptation actions are recorded on the curve.

Figure 35 shows that 10 of the 29 adaptation actions consist of composite service adaptation, shown in red points, while 19 of them are service selections, shown in black points. Figure 36 shows that only 4 of the 29 adaptation actions were for composite service adaptation and the remaining actions were for service adaptation actions.

The figures indicate that local service adaptation actions, especially service substitution,

add more flexibility to the adaptation process. Each predicted failed service can be substituted promptly to maintain the performance of the composite service and avoid a global violation. This minimizes the need for the composite service adaptation action, which has a high computational complexity.

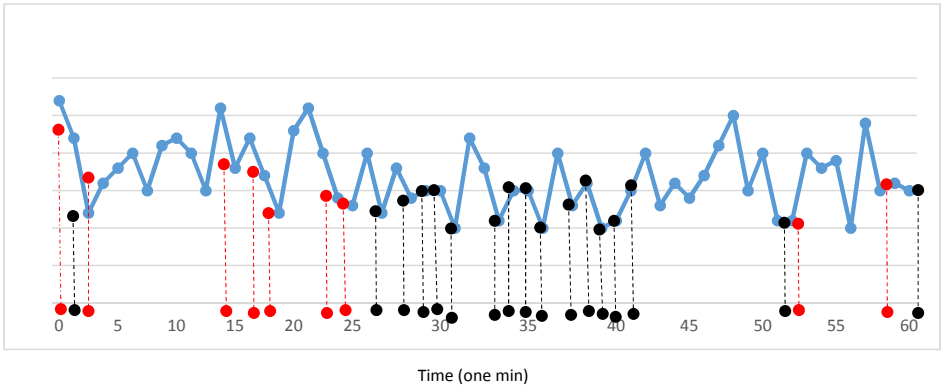


Figure 35: Adaptation flexibility of the centralized approach

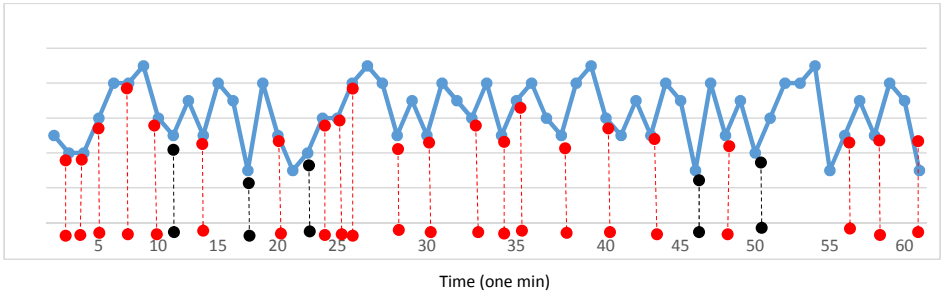


Figure 36: Adaptation flexibility of our approach

5.4 Conclusion

In this chapter, we proposed adaptation actions that enable the master and slaves to secure the composite service and the constitutes. We used the feature model to model the alternative variants of the composite service and we adopted the MAPE loop to manage run-time dynamic adaptation. We modelled the composite service adaptation process as a multi-objective optimization problem to select the optimal feature nodes that maximize the performance and minimize the cost, constrained by SLA and feature model constraints.

In addition, we proposed local service adaptation action that drastically improves performance and reduces overhead by promptly substituting the predicted failed service locally. The experimental results showed that our approach is efficient in dynamic environments compared to a benchmark centralized approach. Particularly, our approach reduces the response time and the cost by 50% and increases the availability by 2%. Moreover, our approach outperforms the benchmark centralized approach by 15% in terms of reducing the execution time in a large-scale environment. In addition, we measured the execution time of our approach where we increased the number of service constituents from 500 to 10000. The results showed that the execution time of our approach grows polynomially with the increase in the number of constituents, which proves the scalability of our approach. Furthermore, local service adaptation action adds to the flexibility of the adaptation process. This decreases the number of composite service adaptation action, which has a high time complexity, by 60%. Moreover, the execution time decreased by 80% when the frequency change is 50%. Thus, this chapter has achieved the fourth research objective (Objective 4) discussed in Chapter 1, which aimed at dynamically adapting composite services running in an open-world against unforeseen context changes.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we addressed the problem of providing context-aware composite services that continue offering their functionalities in dynamic and uncertain running environments. To this end, we designed an adaptation architecture that enables the service to:

- self-configure the workflow and the constituent services to face open world changes;
- self-optimize the QoS performance over run-time by triggering promptly adaptation actions;
- self-heal its failure by the capability to discover, diagnose and react to context changes.
- self-protect its reputation and gained trust against malicious attacks.

In particular, we developed a decentralized adaptation approach that enables the distribution of the whole MAPE loop among multiple mangers, namely the master and the slaves, to manage the composite service adaptation in scalable settings. Experiments conducted on a real services dataset show that our architecture reduces the execution time in a large-scale environment. When we increased the domain size, the results show that the

execution time of our approach grows polynomially with the increase in the number of constituents, which proves the scalability of our approach. In addition, we elaborated on the multi-dimensional service trust model that considers the cyclic QoS dependency relations among QoS metrics and context variables. Promisingly, experiments conducted using the real-life dataset reveal that this solution enhances trust estimation accuracy while being resistant to collusion and white-wash attacks. Moreover, we developed an SLA violation prediction approach that supports decision making by the master regarding composite service adaptation. Experiments prove the prediction accuracy of the proposed solution. This efficiency is because of our approach captures service dependencies in a relational setting, unlike current prediction approaches. Finally, we designed adaptation actions for composite service reconfiguration and constituent services selection and substitution. Experiments show that our approach is efficient in dynamic settings. The proposed constituent service adaptation action adds to the flexibility of the adaptation process by decreasing the need for composite service adaptation action by prompt replacing the failed service.

The following points summarize the main contributions of this thesis:

1. We introduced a two-level adaptation process for composite services that enables the service provider to monitor QoS performance and trigger prompt adaptation actions. More specifically:

    - We applied a master/slaves pattern to manage the composite service adaptation in scalable settings.
    - We distributed the whole MAPE loop among multiple mangers, namely the master and the slaves.

2. We proposed a subjective trust framework for context-aware services that improves the trust prediction accuracy by capturing cyclic dependency relations between QoS metrics and context variables. We modified the dependency network graph to capture

the dynamic cyclic dependency relations linking QoS metrics and context variables.

3. We extended the trust framework to provide a comprehensive and effective trust management for services, including objective and bootstrapping trust. In this contribution:

    - We introduced an objective trust evaluation technique that addresses collusion attacks. The proposed collusion detection method is based on graph theory which is more suitable for sophisticated collusion attacks. Using graph theory, we are able to capture the underlying relationships among referees via their referrals for services.
    - We introduced a trust bootstrapping mechanism for the newcomer services to estimate their initial trust values through behaviour observation based on the concepts of social adoption and ensemble classification. These concepts contribute to our proposed mechanism by making it resilient to the white-washing attacks.

4. We improved SLA violation prediction accuracy by including the runtime contextual environment to address the continuous QoS degradation. In this contribution:

    - We proposed an SLA violation prediction approach based on kalman filters. This approach is efficient for online prediction since it requires small QoS data and rapidly adapts itself towards an accurate prediction of SLA violation. However, it fails to consider QoS dependency that exists in the real business world at run time.
    - We introduced a trust based model for SLA Management capitalizing on the relational dependency network. We leveraged horizontal, vertical, and cyclic QoS dependencies in a relational setting which enhance the SLA violation prediction accuracy.

5. We supported dynamic adaptation by the master by a solution based on the feature model that captures the variability of the composite service. More specifically:

    - We extended the feature model-based service adaptation to represent QoS properties of each feature.
    - We modelled the adaptation process as a multi-objective optimization problem to resolve conflicting goals of minimizing the cost and maximizing the performance of the composite service. The optimization problem takes into account the SLA and feature model constraints.
    - We proposed a genetic-based algorithm to solve the proposed multi-objective optimization problem. The algorithm computes the pareto-optimal set of reconfiguration solutions of composite service.
    - We presented a local service adaptation action to be performed by the slaves which promptly substitute the failed service to maintain the overall performance and reduce the need for global adaptation.

The first contribution is proposed to answer the first research objective (Objective 1), which is to develop an adaptation architecture that overcomes the performance degradation caused by centralized and distributed designs. The second, third, and fourth contributions are proposed to answer the second and third research objectives (Objectives 2 and 3), which are to detect multi-type malicious attacks and support the decision making regarding adaptation actions. Finally, the fifth contribution is proposed to answer the fourth research objective (Objective 4), which is about enabling the service adaptation in a dynamic open world to secure the service performance and avoid economic compensation caused by breach penalties.

6.2 Critical Reflection and Future Work

The research work addressed in this thesis leaves some open questions for future work. In the following, we devise some of these aspects that we plan to address.

Although the proposed adaptation approach is able to predict SLA violation for the composite service using relational dependency network, it fails to comprehensively consider dynamic QoS dependency. The relational dependency network based model (offline trained model) is based on static relational data. Accordingly, it is inefficient for online adaptation because of its incompetence in dynamic environments. Therefore, we plan to reinvestigate it to capture the QoS dependencies between services that interact and operate in environments characterized by continual changes to QoS requirements and/or state of services.

Moreover, context information that represents real-world situations could be inherently imperfect due to, for example, sensor limitations. This information is associated with certain quality indicators, such as precision and freshness [73]. These quality indicators should be considered with context acquisition, aggregation, and reasoning. In addition, context information is privacy sensitive. Users of context-aware services need to control the privacy of their context information. However, users could limit their context information that negatively affects the quality of the adaptation process. A future direction for this thesis is to provide users with control over their privacy with the trade-off between privacy and the quality of service adaptation. We plan to provide a trust-based model that selects trustworthy context information at a specific quality level. We also plan to extend the capability of the context-aware services to avoid privacy violations which is a critical threat to the service reputation.

Bibliography

[1] An architectural blueprint for autonomic computing. Technical report, IBM, June 2005.

[2] G. H. Alférez, V. Pelechano, R. Mazo, C. Salinesi, and D. Diaz. Dynamic adaptation of service compositions with variability models. *Journal of Systems and Software*, 91:24–47, 2014.

[3] G. H. Alférez and Vicente Pelechano. Achieving autonomic web service compositions with models at runtime. *Computers and Electrical Engineering*, 63:332–352, 2017.

[4] Amazon. Amazon EC2 pricing.

[5] Mennatallah Amer and Markus Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conferernce*, 08 2012.

[6] Trosky Arias, Peter Van der Spek, and Paris Avgeriou. A practice-driven systematic review of dependency analysis solutions. *Empirical Software Engineering*, 16:544–586, 10 2011.

[7] Lina Barakat, Simon Miles, and Michael Luck. Adaptive composition in dynamic service environments. *Future Generation Computer Systems*, 80:215–228, 2018.

[8] L. Baresi, E. Di Nitto, and C. Ghezzi. Toward open-world software: Issues and challenges. *Computer*, 39(10):36–43, 2006.

[9] S. Basu, F. Casati, and F. Daniel. Toward web service dependency discovery for soa management. In *2008 IEEE International Conference on Services Computing*, volume 2, pages 422–429, 2008.

[10] David Benavides, Pablo Trinidad Martín-Arroyo, and Antonio Ruiz Cortés. Automated reasoning on feature models. In *Seminal Contributions to Information Systems Engineering*, 2013.

[11] David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortés. Automated reasoning on feature models. In *Proceedings of the 17th International Conference on Advanced Information Systems Engineering*, CAiSE'05, pages 491–503, Berlin, Heidelberg, 2005. Springer-Verlag.

[12] Jamal Bentahar, Zakaria Maamar, Djamal Benslimane, and Philippe Thiran. Using argumentative agents to manage communities of web services. In *21st International Conference on Advanced Information Networking and Applications (AINA 2007), Workshops Proceedings, Volume 2, May 21-23, 2007, Niagara Falls, Canada*, pages 588–593, 2007.

[13] Philip Bianco, Grace Lewis, and Paulo Merson. Service level agreements in service-oriented architecture environments. Technical Report CMU/SEI-2008-TN-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2008.

[14] Jan Bosch. *Design and use of software architectures: adopting and evolving a product-line approach*. Addison-Wesley, 2000.

[15] Antonio Bucchiarone, Martina De Sanctis, and Annapaola Marconi. Decentralized dynamic adaptation for service-based collective adaptive systems. In Khalil Drira,

Hongbing Wang, Qi Yu, Yan Wang, Yuhong Yan, François Charoy, Jan Mendling, Mohamed Mohamed, Zhongjie Wang, and Sami Bhiri, editors, *Service-Oriented Computing – ICSOC 2016 Workshops*, pages 5–20, Cham, 2017. Springer International Publishing.

[16] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola. MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. *IEEE Transactions on Software Engineering*, 38(5):1138–1159, Sept 2012.

[17] B. Chen, X. Peng, Y. Yu, and W. Zhao. Requirements driven self-optimization of composite services using feedback control. *IEEE Transactions on Services Computing*, 8(1):107–120, Jan 2015.

[18] Chien Chin Chen, Yu-Hao Wan, Meng-Chieh Chung, and Yu-Chun Sun. An effective recommendation method for cold start new users using trust and distrust networks. *Information Sciences*, 224:19–36, 2013.

[19] I. Chen, J. Guo, and F. Bao. Trust management for soa-based iot and its application to service composition. *IEEE Transactions on Services Computing*, 9(3):482–495, May 2016.

[20] Y. Chen, J. Huang, C. Lin, and X. Shen. Multi-objective service composition with qos dependencies. *IEEE Transactions on Cloud Computing*, 7(2):537–552, 2019.

[21] Jin-Hee Cho, Ananthram Swami, and Ing-Ray Chen. Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks. *Journal of Network and Computer Applications*, 35(3):1001 – 1012, 2012. Special Issue on Trusted Computing and Communications.

[22] William Conner, Arun Iyengar, Thomas Mikalsen, Isabelle Rouvellou, and Klara Nahrstedt. A trust management framework for service-oriented environments. In

Proceedings of the 18th International Conference on World Wide Web, WWW, pages 891–900. ACM, 2009.

[23] Javier Cubo and Ernesto Pimentel. DAMASCo: A framework for the automatic composition of component-based and service-oriented architectures. In Ivica Crnkovic, Volker Gruhn, and Matthias Book, editors, *Software Architecture*, volume 6903 of *Lecture Notes in Computer Science*, pages 388–404. Springer Berlin Heidelberg, 2011.

[24] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[25] Gianni D'Angelo, Francesco Palmieri, and Salvatore Rampone. Detecting unfair recommendations in trust-based pervasive environments. *Information Sciences*, 486:31–51, 2019.

[26] Danilo Rafael de Lima Cabral and Roberto Souto Maior de Barros. Concept drift detection based on fisher's exact test. *Information Sciences*, 442-443:220 – 234, 2018.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[28] S. Deng, H. Wu, D. Hu, and J. Leon Zhao. Service selection for composition with qos correlations. *IEEE Transactions on Services Computing*, 9(2):291–303, 2016.

[29] Mohamad Eid, Atif Alamri, and Abdulmotaleb El Saddik. A reference model for dynamic web service composition systems. *International Journal of Web and Grid Services*, 4(2):149–168, 2008.

[30] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, 2003.

[31] Xiu-Jun Gong, Shao-Hui Liu, and Zhong-Zhi Shi. An incremental Bayes classification model. *Chinese Journal of Computers*, 25:645–650, 06 2002.

[32] Svein Hallsteinsen, Mike Hinchey, Sooyong Park, and Klaus Schmid. Dynamic software product lines. In Rafael Capilla, Jan Bosch, and Kyo-Chul Kang, editors, *Systems and Software Variability Management: Concepts, Tools and Experiences*, pages 253–260, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[33] C. Hang, A. K. Kalia, and M. P. Singh. Behind the curtain: Service selection via trust in composite services. In *2012 IEEE 19th International Conference on Web Services*, pages 9–16, June 2012.

[34] K. Hashmi, Z. Malik, A. Erradi, and A. Rezgui. Qos dependency modeling for composite systems. *IEEE Transactions on Services Computing*, 11(6):936–947, 2018.

[35] Qiang He, Jun Yan, Hai Jin, and Yun Yang. Adaptation of web service composition based on workflow patterns. In Athman Bouguettaya, Ingolf Krueger, and Tiziana Margaria, editors, *Service-Oriented Computing – ICSOC 2008: 6th International Conference, Sydney, Australia, December 1-5, 2008. Proceedings*, pages 22–37, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[36] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75, September 2001.

[37] Geoff Hulten, David Maxwell Chickering, and David Heckerman. Learning Bayesian networks from dependency networks: A preliminary study. In *Proceedings of the*

Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS, Key West, Florida, USA, January 3-6, 2003.

[38] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, page 97–106, New York, NY, USA, 2001. Association for Computing Machinery.

[39] M. C. Jaeger, G. Rojec-Goldmann, and G. Muhl. Qos aggregation for web service composition using workflow patterns. In *Proceedings. Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004. EDOC 2004.*, pages 149–159, Sep. 2004.

[40] J. Jang, D. Shin, and K. Lee. Fast quality driven selection of composite web services. In *2006 European Conference on Web Services (ECOWS'06)*, pages 87–98, Dec 2006.

[41] C. Jatoth, G. R. Gangadharan, and R. Buyya. Computational intelligence based qos-aware web service composition: A systematic literature review. *IEEE Transactions on Services Computing*, 10(3):475–492, May 2017.

[42] Deng Ju-Long. Control problems of grey systems. *Systems & Control Letters*, 1(5):288 – 294, 1982.

[43] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM.

[44] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer

Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical Report Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1990.

[45] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, Jan 2003.

[46] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[47] Daphne Koller, Uri Lerner, and Dragomir Anguelov. A general algorithm for approximate inference and its application to hybrid bayes nets. *UAI*, 15, 01 2013.

[48] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E. J. Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3):42–51, July 2003.

[49] Sotiris Kotsiantis. Increasing the accuracy of incremental naive Bayes classifier using instance based learning. *International Journal of Control, Automation and Systems*, 11(1):159–166, 2013.

[50] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar. Monitoring, prediction and prevention of sla violations in composite services. In *2010 IEEE International Conference on Web Services*, pages 369–376, 2010.

[51] G. S. Li and N. Wang. Web service qos prediction with adaptive calibration. In *2015 International Conference on Computer Science and Applications (CSA)*, pages 351–356, Nov 2015.

[52] Li Li, Pengyi Yang, Ling Ou, Zili Zhang, and Peng Cheng. Genetic algorithm-based multi-objective optimisation for qos-aware web services composition. In Yaxin Bi and Mary-Anne Williams, editors, *Knowledge Science, Engineering and Management*, pages 549–554, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[53] Helan Liang, Yanhua Du, Ting Jiang, and Fanzhang Li. A comprehensive multi-objective approach of service selection for service processes with twofold restrictions. *Future Generation Computer Systems*, 92:119 – 140, 2019.

[54] Thibaut Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *CoRR*, abs/1007.4063, 2010.

[55] Zaki Malik and Athman Bouguettaya. Rateweb: Reputation assessment for trust establishment among web services. *The VLDB Journal*, 18(4):885–911, Aug 2009.

[56] S. Manocha and Mark A. Girolami. An empirical analysis of the probabilistic k-nearest neighbour classifier. *Pattern Recognition Letters*, 28(13):1818–1824, 2007.

[57] M. Mehdi, N. Bouguila, and J. Bentahar. A qos-based trust approach for service selection and composition via bayesian networks. In *2013 IEEE 20th International Conference on Web Services*, pages 211–218, June 2013.

[58] M. Mehdi, N. Bouguila, and J. Bentahar. Trust and reputation of web services through qos correlation lens. *IEEE Transactions on Services Computing*, 9(6):968–981, Nov 2016.

[59] M. Motallebi, F. Ishikawa, and S. Honiden. Trust computation in web service compositions using bayesian networks. In *2012 IEEE 19th International Conference on Web Services*, pages 623–625, June 2012.

[60] A. Moustafa, M. Zhang, and Q. Bai. Trustworthy stigmergic service compositionand adaptation in decentralized environments. *IEEE Transactions on Services Computing*, 9(2):317–329, March 2016.

[61] R. Neisse. *Trust and privacy management support for context-aware service platforms*. PhD thesis, University of Twente, Netherlands, 3 2012. SIKS Dissertation Series No. 2012-09.

[62] Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 03 2007.

[63] H. T. Nguyen, W. Zhao, and J. Yang. A trust and reputation model based on bayesian network for web services. In *2010 IEEE International Conference on Web Services*, pages 251–258, July 2010.

[64] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In Vincent Corruble, Masayuki Takeda, and Einoshin Suzuki, editors, *Discovery Science*, pages 264–269, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[65] D. L. Parnas. Designing software for ease of extension and contraction. *IEEE Transactions on Software Engineering*, SE-5(2):128–138, 1979.

[66] Barbara Pernici and S. Hossein Siadat. Adaptation of web services based on QoS satisfaction. In E. Michael Maximilien, Gustavo Rossi, Soe-Tsyr Yuan, Heiko Ludwig, and Marcelo Fantinato, editors, *Service-Oriented Computing: ICSOC 2010 International Workshops, PAASC, WESOA, SEE, and SOC-LOG, San Francisco, CA, USA, December 7-10, 2010, Revised Selected Papers*, pages 65–75, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[67] Ali Asghar Pourhaji Kazem, Hossein Pedram, and Hassan Abolhassani. Bnqm: A bayesian network based qos model for grid service composition. *Expert Systems with Applications*, 42(20):6828 – 6843, 2015.

[68] V. R. Puttige and S. G. Anavatti. Comparison of real-time online and offline neural network models for a uav. In *2007 International Joint Conference on Neural Networks*, pages 412–417, Aug 2007.

[69] Irma Ravkic, Jan Ramon, and Jesse Davis. Learning relational dependency networks in hybrid domains. *Machine Learning*, 100, 05 2015.

[70] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264. Curran Associates, Inc., 2007.

[71] Mark Senn. *WS-DREAM: Towards open datasets and source code for web service research*.

[72] Qiping She, Xiaochao Wei, Guihua Nie, and Donglin Chen. QoS-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence. *Expert Systems with Applications*, 138:112804, 2019.

[73] K. Sheikh, M. Wegdam, and M. van Sinderen. Middleware support for quality of context in pervasive context-aware systems. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07)*, pages 461–466, March 2007.

[74] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[75] Claudio De Stefano, Antonio Della Cioppa, and Angelo Marcelli. An adaptive weighted majority vote rule for combining multiple classifiers. In *16th International Conference on Pattern Recognition, ICPR, Quebec, Canada, August 11-15*, pages 192–195, 2002.

[76] B. Tang and M. Tang. Bayesian model-based prediction of service level agreement violations for cloud services. In *2014 Theoretical Aspects of Software Engineering Conference*, pages 170–176, 2014.

[77] Raquel Ureña, Gang Kou, Yucheng Dong, Francisco Chiclana, and Enrique Herrera-Viedma. A review on trust propagation and opinion dynamics in social networks and group decision making frameworks. *Information Sciences*, 478:461–475, 2019.

[78] A. Vogel, B. Kerherve, G. von Bochmann, and J. Gecsei. Distributed multimedia and qos: a survey. *IEEE MultiMedia*, 2(2):10–19, 1995.

[79] L. Vu and K. Aberer. Towards probabilistic estimation of quality of online services. In *2009 IEEE International Conference on Web Services*, pages 99–106, July 2009.

[80] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. A survey on trust and reputation models for web services: Single, composite, and communities. *Decision Support Systems*, 74:121 – 134, 2015.

[81] Omar Abdel Wahab, Jamal Bentahar, Hadi Otrok, and Azzam Mourad. Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game. *IEEE Trans. Services Computing*, 11(1):184–201, 2018.

[82] C. Wang and J. L. Pazat. A two-phase online prediction approach for accurate and timely adaptation decision. In *2012 IEEE Ninth International Conference on Services Computing*, pages 218–225, June 2012.

[83] H. Wang, C. Yu, L. Wang, and Q. Yu. Effective bigdata-space service selection over trust and heterogeneous qos preferences. *IEEE Transactions on Services Computing*, 11(4):644–657, July 2018.

[84] Shangguang Wang, Zibin Zheng, Zhengping Wu, Michael R. Lyu, and Fangchun Yang. Reputation measurement and malicious feedback rating prevention in web service recommendation systems. *IEEE Trans. Services Computing*, 8(5):755–767, 2015.

[85] Xiaofeng Wang, Ling Liu, and Jinshu Su. RLM: A general model for trust representation and aggregation. *IEEE Trans. Services Computing*, 5(1):131–143, 2012.

[86] Y. Wang, I. Chen, J. Cho, A. Swami, Y. Lu, C. Lu, and J. J. P. Tsai. Catrust: Context-aware trust management for service-oriented ad hoc networks. *IEEE Transactions on Services Computing*, 11(6):908–921, Nov 2018.

[87] Y. Wang and J. Vassileva. Bayesian network-based trust model. In *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 372–378, Oct 2003.

[88] Zhuo Wang, Songmin Gu, Xiangnan Zhao, and Xiaowei Xu. Graph-based review spammer group detection. *Knowledge and Information Systems*, 55(3):571–597, Jun 2018.

[89] Guiyi Wei, Yun Ling, Binfeng Guo, Bin Xiao, and Athanasios V. Vasilakos. Prediction-based data aggregation in wireless sensor networks: Combining grey model and kalman filter. *Computer Communications*, 34(6):793 – 802, 2011.

[90] Jules White, Harrison D Strowd, and Douglas C Schmidt. Creating self-healing service compositions with feature models and microrebooting. *International Journal of Business Process Integration and Management*, 4(1):35–46, 2009.

[91] Matthias Winkler, Thomas Springer, Edmundo David Trigos, and Alexander Schill. Analysing dependencies in service compositions. In Asit Dan, Frédéric Gittler, and Farouk Toumani, editors, *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pages 123–133, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[92] Li Xiong and Ling Liu. Peertrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, July 2004.

[93] Hamdi Yahyaoui and Sami Zhioua. Bootstrapping trust of web services based on trust patterns and hidden markov models. *Knowledge and Information Systems*, 37(2):389–416, 2013.

[94] Ying Yang and Geoffrey I. Webb. Non-disjoint discretization for naive-Bayes classifiers. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML), Sydney, Australia, July 8-12*, pages 666–673, 2002.

[95] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Trans. Web*, 1(1), 2007.

[96] Valarie A. Zeithaml. Consumer perceptions of price, quality, and value: A means-end model and synthesis of evidence. *Journal of Marketing*, 52(3):2–22, 1988.

[97] Liangzhao Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[98] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, 6(3):289–299, July 2013.

[99] J. Zhu, P. He, Z. Zheng, and M. R. Lyu. Online QoS prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):2911–2924, Oct 2017.