Establishment and Maintenance of Trustworthy Composite Services Using Multidimensional Service Attributes

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Hisain Elshaafi

Department of Computing, Mathematics & Physics
Waterford Institute of Technology

Supervisor: Dr. Dmitri Botvich

Co-supervisor: Mr. Jimmy McGibney

Thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

Submitted to Waterford Institute of Technology, October 2014

Declaration

I, Hisain Elshaafi (Student ID: 20021505), declare that this thesis titled, "Establishment and Maintenance of Trustworthy Composite Services Using Multidimensional Service Attributes" and the work presented in it are my own. I confirm that no part of the thesis has been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ Date: _____

Establishment and Maintenance of Trustworthy Composite Services Using Multidimensional Service Attributes

Hisain Elshaafi

Abstract

Service composition is a key concept in Service Oriented Computing. Distributed low level services are assembled in business processes providing enterprise services. The openness, distribution, dynamicity and other characteristics of service compositions and of service environments in general, present challenges to the widespread adoption by industries and end-users. Those characteristics result in difficulties in establishing and building trust between different entities. Composition techniques must be able to measure trustworthiness and identify trustworthy component services. Composition techniques also must be able to maintain trustworthiness of composite services against possible dynamic changes in the environment or the services themselves such as changes in demand or service behaviour.

This research views trustworthiness as a multidimensional concept that reflects an extensible set of attributes of services with no single aspect, such as identity verification, is sufficient to ensure trustworthiness. This thesis investigates the state of the art in the related areas and describes the background and context of the work. It details mechanisms for the aggregation and calculation of trustworthiness of composite services based on composition plans and a range of trustworthiness attributes. The thesis also describes prediction techniques for service trustworthiness based on its monitoring. The research describes a framework for measuring security aspects of services and their composition based on protection methods and the exploitability of the service-related resources. Furthermore, it describes methods for determining trustworthiness of component services from that of composite services. Since operational issues are important in maintaining and balancing composite service trustworthiness and profitability, the thesis describes resource management techniques as well as an approach to the optimisation of pricing that contribute to achieving those goals.

The overall goal of this research is to provide the techniques for establishing and maintaining trustworthy composite services where sets of component services each with a distinct functionality are offered by multiple component providers. Alternative component services are available for selection for each task in a composition. Those alternative services may have different values of their attributes as well as different patterns of attribute changes over time.

Acknowledgements

Special thanks to my PhD supervisor Dmitri Botvich for his continued help and exceptional guidance in this study. His deep insights helped me at every stage of my research. Many thanks also to Jimmy McGibney for his support and technical advice.

I would like to thank all my family and friends who supported and encouraged me during this endeavour from start to end.

Thanks to TSSG who supported this research through valuable resources and excellent learning environment. I am grateful to everyone who offered reviews, feedback and general support.

The PhD research has received funding from the EU Seventh Framework Programme (FP7/2007-2013) under grant no. 257930 (Aniketos). Aniketos is an EU research project that addresses trustworthy and secure service compositions with run-time monitoring and adaptation of services. I would like to thank academic and industrial partners in this project who were engaged in discussions on related topics and offered feedback to this work.

Contents

List of Figures

List of Tables

Abbreviations

ASCE	Adelard Assurance and Safety Case Environment
BoD	Binding of Duty
BOF	Buffer Overflow
BPEL/WS-BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
CS	Composite Service
CSP	Composite Service Provider
CSRF	Cross-Site Request Forgery
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DHT	Distributed Hash Table
DoS	Denial of Service
GA	Genetic Algorithm
HTTP	Hypertext Transfer Protocol
LDAP	Lightweight Directory Access Protocol
MCDM	Multiple Criteria Decision Making
MDA	Model Driven Architecture
MNL	Multinomial Logit
OMG	Object Management Group Standards Consortium
OSGi	Open Services Gateway initiative framework
OWASP	Open Web Application Security Project
P2P	Peer to Peer

PEP	Policy Enforcement Point
QoE	Quality of Experience
QoS	Quality of Service
QoWS	Quality of Web Service
RSA	Public-key cryptography algorithm named after its authors; Rivest, Shamir and Adleman
SACM	Structured Assurance Case Model
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SoD	Separation of Duty
TCG	Trusted Computing Group
TM	Trust Management
TwC	Trustworthy Computing
WoT	Web of Trust
WS	Web Service
WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language
XSS	Cross Site Scripting

Chapter 1

Introduction

1.1 Background

Service Oriented Computing (SOC) is increasingly popular, with increased attention from industry. A key concept is that services can be dynamically or statically composed to create new higher level services. The composition of services has several benefits such as creating new business opportunities, flexible use of resources and optimising operations within and across organisations. A service in a composition is called *component service*. A component service may be an atomic or composite service. An atomic service is indivisible and it encapsulates one or few functionalities.

Services are described, published, discovered, and assembled, providing distributed business processes exposed externally as composite services. A *composite service (CS)* may be used directly by an end-user or recursively incorporated into further service compositions. Orchestration and choreography refer to two approaches to service composition. Orchestration indicates control from one party over the business process. Choreography, on the other hand, is a more decentralised process where each component service acts as a peer and the approach requires each party to describe and track its part in the interactions.

The distribution, dynamicity and other characteristics of service compositions, and of service environments in general, present challenges to the widespread adoption by industries and end-users. This is due to the consequent lack of basis for establishing and building trustworthiness between different parties based on first-hand knowledge.

For a composite service to be trusted by its target consumers, service composition techniques must be able to identify which component services are trustworthy. The composition techniques also must be able to maintain the most trustworthy and cost efficient composite service. Maintaining trustworthiness helps provide a safe environment for businesses and consumers to dynamically interact and carry out transactions.

Trustworthy service compositions additionally require runtime monitoring and the capability for service adaptation. The adaptations are needed due to changing operational, business or threat environments or due to changes in service qualities and behaviour. Among the challenges to addressing those problems is that the changes may not be easily predictable. Since down-time is costly, a CS must be able to operate even during an attack or increased demand, taking risks and adaptation costs into account.

The overall goal of the thesis is to provide the techniques for establishing and maintaining trustworthy composite services where sets of component services each with a distinct functionality are offered by multiple component providers. Alternative services are available for selection for each task in a composition. Those alternative component services may have different values of their attributes as well as different patterns of attribute changes over time.

1.2 Structure of the Document

The thesis is organised into following chapters.

Chapter 2 describes the research problem and questions that addressed in the thesis. The chapter also discusses the scope and outlines the contributions of this research. A common motivating scenario described in this chapter serves to integrate the purpose

and objectives of the research. A list of publications based on the research is included in Section 2.7.

Chapter 3 investigates current and recent research work and the state of the art in areas related to this work such as reputation management, trust, security attributes, *Quality of Service (QoS)*, resource management and service profitability. The discussion of the related works investigates their relevance to this research, their contributions and shortcomings.

The following chapters are the detailed contributions of this PhD research.

Chapter 4 presents an approach to the monitoring and prediction of trustworthiness of services that are assembled from lower level component services. The chapter discusses the aggregation of trustworthiness attributes for a composite or component service into a common trustworthiness value. It also describes a trustworthiness monitoring and prediction software module that processes data about services and uses it to predict trustworthiness of CSs.

Chapter 5 describes an attack surface based security metrics framework for measuring security of services and their composition through evaluating protection methods and the exploitability of service's resources (i.e. its operations and data). The computation of the attack surface metric depends on the types and the number of service resources and the level of compliance of the service to the optimum values of the applicable security aspects and attributes.

Chapter 6 details a novel framework and mechanisms for the determination of some important trustworthiness attributes of components namely; reliability, reputation, response time and capacity. The determination is based on monitoring and consumer reporting of the corresponding attributes of composite services. The distributed component services are jointly invoked in multiple collaborating composite services.

Chapter 7 addresses enhancing profitability of providers of composite services through novel approaches to the related issues of competitive CS pricing, cost efficiency and trustworthiness.

Chapter 8 concludes the thesis by discussing the overall findings and the challenges faced. It also describes recommendations for future work.

Chapter 2

Research Problem and Contributions

2.1 Problem Statement

In service environments, services interact with each other and with their end users. Some of the important characteristics of service environments are that they are dynamic, distributed, loosely coupled and open. These characteristics result in the possible existence of various levels of security, availability, reliability, capacity, and other attributes of the services operating in such environments. Consequently, it creates challenges for the interacting services and service consumers that want to only carry out transactions with services that are trustworthy. A central concept in the services paradigm is that services can be composed to create new enterprise services. For a composite service to be trustworthy for utilisation by its target consumers, and for services to be able to trust and rely on other possibly unknown services, service composition techniques must be able to identify which component services are trustworthy. The composition techniques also must be able to maintain the most trustworthy and cost efficient composite service. Maintaining trustworthiness helps consumer confidence and provides a safe environment for businesses to dynamically interact and carry out transactions. Therefore, addressing establishment and maintenance of multidimensional trust is essential for the success and adoption of the services paradigm.

2.2 Research Questions

Based on the research problem described in Section 2.1, the following Research Questions (RQs) outline the main challenges towards the solution.

(i) *How can attributes of a service, that can affect its trustworthiness, be aggregated when composing a service?*

Work on this RQ is included as part of Chapter 4 and published mainly in [1].

(ii) *How can the trustworthiness be predicted for a composite service based on the monitored attributes of its components? What architecture and algorithms could be used to support the monitoring, prediction and optimisation of its trustworthiness?*

Work on this RQ is included as part of Chapter 4 and published mainly in [2].

(iii) *How can the security level of a service be quantified in order to be considered in predicting service trustworthiness?*

Work on this RQ is included in Chapter 5 and currently submitted for publication [3].

(iv) *How can the trustworthiness of components be determined from that of operating composite services when attribute metrics could be received on the operation of the compositions?*

Work on this RQ is included in Chapter 6, most are published [4, 5]. Part of the work is planned for submission for publication.

(v) *How can the profitability of composite service providers be enhanced by exploiting the ability to predict the trustworthiness of composite services?*

Work on this RQ is included in Chapter 7 and published mainly in [6].

See also Section 2.7 for the list of the author's publications and current submitted articles.

2.3 Scope

This PhD research focuses on the techniques that are required to establish and maintain trustworthy composite services. The work considers trust and trustworthiness as multifaceted concepts that reflect an extensible set of attributes of services. Security mechanisms such as encryption, authentication, and authorization are necessary steps in establishing trust. For example, authentication mechanisms assure the consumer that the service provider is who it claims it is. However, the service may not behave in the way it is required or expected in terms of reliability, availability, reputation, etc. Therefore, security mechanisms are not sufficient as they do not completely assure the behaviour of the service.

Several research works consider trust as multifaceted concept [e.g. 7–13]. Initial trust definitions by researchers were limited to certain aspects of trust such as the definition by Gambetta [14] which was focused on reliability of an agent. Grandison and Sloman [15] define trust and identify classes of trust including service provision trust. Jøsang et al. [11, 16] define provision trust as *"the relying party's trust in a service or resource provider, and is relevant when the relying party is a user seeking protection from malicious or unreliable service providers"*. Some researchers such as Chang et al. [12] and Artz and Gil [13] seek to improve the definition of trust by identifying shortcomings of definitions by previous authors. More discussion on the concept of trust and its definition in Section 3.2.1. Our definition of trust aims to include the key aspects mentioned in the definitions by these authors including context, expectation, reliance, action and outcome. We define *trust, trustworthiness, trustworthiness attribute and reputation* as follows.

- *Trust* is a relationship between two or more entities that indicates the context-based expectations from an entity towards another in relation to reliance in accomplishing a certain action at a certain quality or outcome.
- *Trustworthiness* of an entity is a value representing the level of trust that the trusting entity or its agent has in that entity.

- *Trustworthiness Attribute* is a measurable attribute of a service whose variation in value can affect the objective and/or perceived trustworthiness of the service. A value for an attribute (referred to as *attribute value*) can be determined by aggregating subjective consumer ratings or objective metrics e.g. using automated monitoring systems.

- *Reputation* is the information available about an entity based on consumer ratings that can be used together with other types of information in determining the entity's trustworthiness.

Note that we refer also to *trustworthiness attribute ratings* in the context of the trustworthiness software module. A trustworthiness attribute rating is a value generated by the trustworthiness module's rules engine to normalise metrics and consumer ratings. The trust engine uses the trustworthiness attribute ratings related to a service to compute its trustworthiness value.

The research also includes a business view that aims to justify the consideration of trustworthiness in the business context. The business view is important because a composite service is in effect a business process that aims to accomplish sustained profit and build business value for its providers. Part of this work also investigates directly related concepts and techniques that can contribute to composite service trustworthiness such as resource management and control.

Some research areas, such as those investigating methods of service compositions, may be discussed in the literature review as part of the background to this research although they may not be within the scope described. The aim of the discussion of those areas is to provide the context of this research and its applicability and positioning within the existing technical specifications, standards and established environments.

Although the research aims to develop robust trustworthiness aggregation and prediction models and algorithms, it does not intend to develop new protection mechanisms against some types of attacks such as slandering and dishonest ratings. There have been

advances in protection against such well-understood attacks on trust and reputation systems. The existing protection techniques are utilised in developing the algorithms such as the trustworthiness update (see Chapter 4).

In our work, we evaluate security attributes as subset of trustworthiness attributes. The model-based framework we develop in the thesis, allows recording reported evidence that supports claims about security attributes. Our work focuses on internal characteristics of services. It needs to evaluate exploitability of resources belonging to a service and their defence methods regardless of specific threats or known vulnerabilities. Nonetheless, detection of threats and vulnerabilities can serve as an extension to our approach. Therefore, we consider threat updates as an interesting and significant future work as we describe in Chapter 8.

Third party certification is often considered in trust management [e.g. 18]. However, in the view of the author of the thesis, certification is a reference that helps assure provider's trustworthiness for consumers rather than an actual attribute of a service. Therefore, we consider certification to be outside the scope of this thesis. However, certification can be used as evidence of claims representing security attributes as described above.

In the thesis we examine QoS as one of the categories of trustworthiness attributes as it clearly affects trustworthiness. We consider QoS as objective evaluations of the status of a service from a number of aspects called trustworthiness attributes such as reliability, availability and response time.

2.4 Service Composition Constructs

Component services are executed in a *BPMN* (*Business Process Model and Notation* [19]) business process which is viewed externally as a Web service. A BPMN business process consists of one or more path constructs. Each construct contains one or more service activities. A component service is selected for each activity. The following are common constructs (illustrated in Figure 2.1):

- *Sequence:* Services are invoked one after another.
- *Synchronized Parallel (AND split/AND join):* Two or more services are invoked in parallel and their outcome is synchronized. All services must be executed successfully for the next activity (service) to be executed.
- *Loop or Iteration:* A service is invoked in a loop until a condition is met. We assume that the number of iterations or its average is known.
- *Exclusive Choice (XOR Split/XOR join):* A service is invoked instead of others if a condition is met. We assume that the likelihood of each alternative service to be invoked is known.
- *Unsynchronized Parallel (AND split/OR join):* Two or more services are executed in parallel but no synchronization of the outcome of their execution. The next activity can commence as soon as one service is completed.
- *Multi-choice with Synchronized Merge (OR split/AND join):* Multiple services may be executed in parallel. Subsequent services can begin execution when all executing branches are completed. In BPMN, inclusive gateways are used to split and merge the process flow.
- *Unordered Sequence:* Multiple services are executed sequentially but arbitrarily.

We use θ to denote a service construct in a composition. In BPMN, AND join/split gateway is signified with '$+$', OR with '$\circ$' and XOR with '$\times$'. An empty gateway '$\diamond$' means it waits for one incoming branch before triggering the outgoing flow. We use the empty gateway in merging Unsynchronised Parallel paths. *Inclusive* gateways '$\circledcirc$' are used to split and merge the process flow in Multi-Choice with Synchronized Merge.

It should be noted that for the purpose of trustworthiness attribute aggregation (Section 4.2.1), a composite service represents a hierarchy of constructs. See Subsection 4.2.3 for more details together with an illustrative example.

The constructs covered in this thesis are by far the most common in business processes. Other constructs are much less used in practice and less supported by software

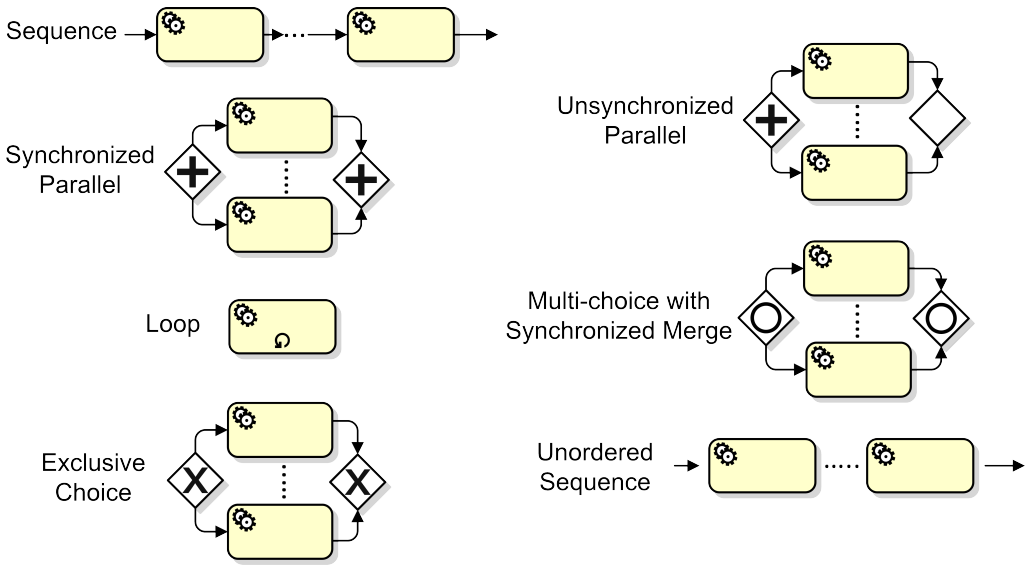


Figure 2.1: Common Service Composition Constructs

tools. Therefore, the approach applies in general to composite services. Where a composite service contains an unusual construct, the approach can be extended with other aggregation techniques that would depend on the characteristics of the construct. Less common and more complex constructs and patterns are supported by modelling languages and products to varying degrees. The structure of business processes including BPMN-based business processes are described by researchers [e.g. 20–22].

2.5 Common Motivating Scenario

Figure 2.2 shows an illustrative example using BPMN notations of a set of composite services in an e-commerce product purchasing scenario. Each composite service is offered by a single *Composite Service Providers (CSPs)*. They contain twelve abstract component services providing for the various activities in the workflows with most of the components shared between two or three composite services. The component services are invoked in the business processes that consist of constructs as indicated in the figure including Sequence, Synchronised Parallel and Exclusive Choice. The service labels are generally self-explanatory. The composite service *Service1* supports bidding on prices and debit payment while *Service2* supports product insurance and only credit

payments are possible. *Service3* consumer can choose to buy by credit payment service. Credit payment requires applying for a loan and checking for consumer credit history and loan rate. Alternatively, the consumer can purchase the product through a debit payment.

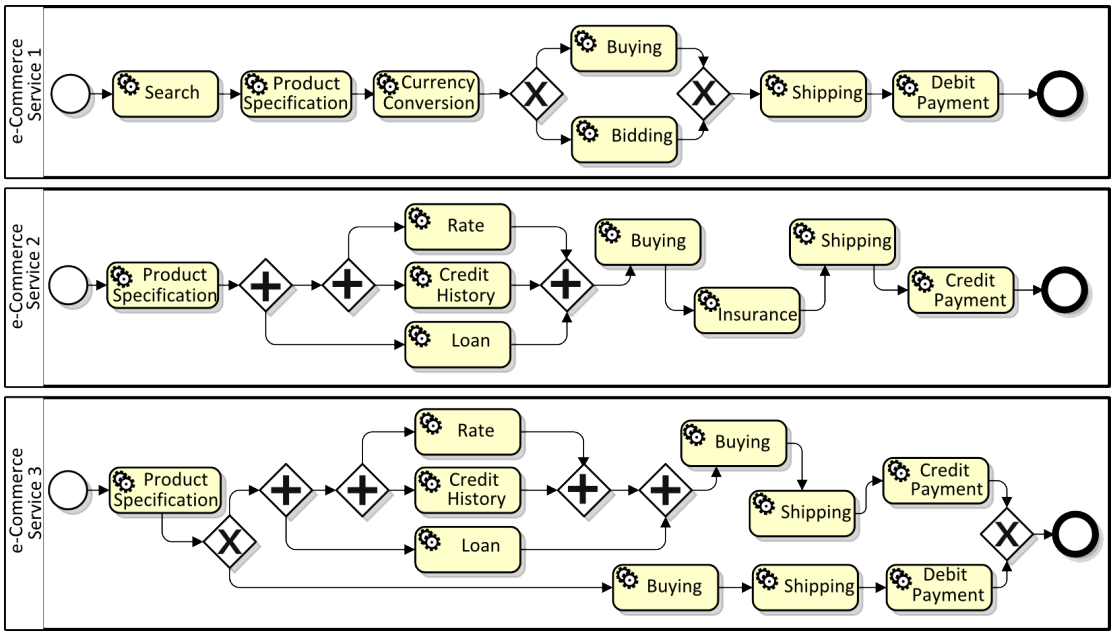


Figure 2.2: Example Scenario of e-Commerce Composite Services Sharing Distributed Components

Each composite service interacts with consumers as a whole service and the component providers are invisible to the consumers. The services are implemented using a BPMN process with a Web portal for the consumers that collects their data and submits them to the BPMN execution engine for processing the orders. The consumers are incentivised to submit a straightforward *Quality of Experience (QoE)* rating through emphasizing their importance in improving the service and ensuring their continued satisfaction and trust.

Consumer QoE reports reflect their perception on service trustworthiness attributes to varying degrees usually with reliability, responsiveness and security being the main attributes that influence the perception [e.g. 23]. However, in this thesis we focus on satisfaction based on consumer's perception of service reliability which, as several research studies indicate, has a major role in trustworthiness and consumer satisfaction [23–26].

A BPMN engine such as the open-source Java-based Activiti [27] engine is extended to provide interfaces for monitoring composite service reliability and response time and a portal to receive QoE reports from consumers. Several techniques have been proposed by researchers on extending BPMN and process execution environments to support non-functional attributes such as Saeedi et al. [28], Pavlovski and Zou [29] and Moser et al. [30].

The composite services can jointly invoke some common distributed component services. The CSPs aim to select the most trustworthy component services available in the distributed service environment that can carry out the composite service activities. As discussed throughout the thesis, the overall trustworthiness of a service is determined based on a set of trustworthiness attributes. The identification of a set of attributes suitable for a particular business activity or for a service environment depends on context, requirements and possibly other factors. The CSPs in this example consider a set of common trustworthiness attributes that can affect the reputation, QoS and security of their CSs (Subsection 4.2.1 discussed the required set of attributes).

Each CSP can select component services from a pool of services some of which offer the same functionality but with varying costs and trustworthiness. Since a trustworthy service must be secure, the CSPs require evaluating the security of the components and taking security into account in the prediction of trustworthiness during component service selection. The goal of each CSP is to optimise its profitability from the composite services through optimal management of trustworthiness, pricing and cost.

Once the components are selected the CSPs require maintaining the trustworthiness of the composite service during its operation. The CSPs can adapt their composite services if an unfavourable change to a component service occurs by replacing it with another. Adaptation may be required as a result of:

- Changes in the trustworthiness based on a runtime change in attributes of the services,
- Changes in the trustworthiness requirements,

– Changes in service environment e.g. new threats, or

– The emergence of new services that are more trustworthy.

However, the CSPs provide measures to maintain the trustworthiness of the component services through determining the required component capacities during executions and management of request admissions.

The CSPs need to determine the values of component trustworthiness attributes based on those of the composite services. The detected component attribute values are used by the CSPs to improve their composite service trustworthiness through service adaptation by replacing untrustworthy components [30]. The attribute values are also used for ranking and selection of components for new CSs.

2.6 Contributions

The following describe the contributions of this thesis to advance the state of the art.

- *Attribute Aggregation*: Aggregation models for service trustworthiness attributes are developed based on composite service structure and its characteristics. The models consider the importance of component services, the probability of executions and business process models which are not considered in existing work. The approach taken in the aggregation protects from unaddressed threats such as the presence of an untrustworthy component in the composition. See Chapter 4 for more details.

- *Trustworthiness Aggregation*: An algorithm for the aggregation of trustworthiness attributes of a composite or component service into a common trustworthiness value. See Chapter 4 for more details.

- *Trustworthiness Update Algorithm*: Trustworthiness prediction algorithm for services. The fast algorithm allows trustworthiness to be updated continuously based

on new ratings/metrics received and on the recency of past ratings. See Chapter 4 for more details.

- *Composition Optimisation Algorithm*: Optimisation of service composition using a custom *Genetic Algorithm (GA)* to determine the optimal set of component services based on their trustworthiness and cost. See Chapter 4 for more details.

- *Trustworthiness Modular Architecture*: Trustworthiness software module with a novel modularised architecture for the implementation of monitoring and prediction techniques and algorithms. The module includes a rules engine for service rating and a trust engine that calculates and updates the trustworthiness values based on the ratings. See Chapter 4 for more details.

- *Security Metric Framework of Services based on Attack Surface Concept*: The application of the concept of attack surfaces to service oriented computing. In particular, it contributes the composition of attack surfaces of component services together with the attack surfaces of their business processes. The research also contributes in taking a multifaceted view of the concept of attack surfaces and the classification of security aspects of services based on the protection methods and the exploitability of the service resources (i.e. its operations and data). Another contribution is the use of the *Structured Assurance Case Model (SACM)* [31, 32] in measuring the security attributes of resources belonging to entry and exit points of components and their re-evaluation as components are incorporated into service compositions. See Chapter 5 for more details.

- *Determination of Component Trustworthiness from Collaborating Compositions*: A framework and a detailed approach for the determination of component trustworthiness based on trustworthiness attributes of composite services. The approach can identify untrustworthy components and detect the trustworthiness of candidate components to be selected for new composite services. The composite services jointly invoke distributed component services. The CSs can have varying degrees of similarity in terms of their components and they may have

different process structure. The degree of similarity and distribution of components between collaborating CSs influence the certainty and preciseness of the predictions. See Chapter 6 for more details.

- *Trustworthiness-related Support Techniques for Profitability*: Business aspects of composite service trustworthiness including; optimisation of prices of composite services based on price response function and trustworthiness, capacity dependent charging and profitability issues e.g. consumer differentiation. See Chapter 7 for more details.

- *Resource Management Integrations with Trustworthiness*: An initial model of trust-oriented resource management. The feedback of resource management to the trustworthiness module in relation to the dynamic capacity of components and the role of resource management in maintaining trustworthiness. The described techniques include priority-based capacity determination and admission including shared resources, CS priorities, consumer differentiation and capacity adjustment. See Chapter 7 for more details.

2.7 List of Publications and Current Submissions

The following are the published and submitted articles in conferences and journals during the course of this research in chronological order.

(i) H. Elshaafi, J. McGibney and D. Botvich, "Trustworthiness Monitoring of Dynamic Service Compositions," *Proc. 6th Workshop on Enhanced Web Service Technologies (WEWST)*, ACM, pp. 25-29, Lugano, Switzerland, Sep 2011, doi: 10.1145/2031325.2031329 [33].

(ii) H. Elshaafi, J. McGibney and D. Botvich, "Business Driven Optimisation of Service Compositions," *Proc. 7th Int. Conf. Next Generation Web Services Practices (NWeSP)*, IEEE, pp. 119-124, Salamanca, Spain, Oct 2011, doi:10.1109/NWeSP.2011.6088163 [34].

(iii) H. Elshaafi, J. McGibney and D. Botvich, "Trustworthiness Monitoring and Prediction of Composite Services," *Proc. 17th IEEE Symposium on Computers and Communication (ISCC)*, pp. 580-587, Cappadocia, Turkey, Jul 2012, doi: 10.1109/ISCC.2012.6249359 (*best paper award*) [2].

(iv) H. Elshaafi and D. Botvich, "Aggregation of Trustworthiness Properties of BPMN-based Composite Services," *Proc. 17th IEEE Int. Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (CAMAD)*, pp. 383-387, Barcelona, Spain, Sep 2012, doi:10.1109/CAMAD.2012.6335373 [1].

(v) H. Elshaafi, J. McGibney and D. Botvich, "Profitability and Cost Management of Trustworthy Composite Services," *Proc. 9th Int. Conf. Trust, Privacy & Security in Digital Business (TrustBus)*, Springer, pp. 179-191, Vienna, Austria, Sep 2012, doi:10.1007/978-3-642-32287-7_16 [6].

(vi) H. Elshaafi and D. Botvich, "Trustworthiness Inference of Multi-tenant Component Services in Service Compositions," *Proc. 4th FTRA Int. Conf. Computer Science and its Applications (CSA)*, Springer, pp. 301-312, Jeju, Korea, Nov 2012, doi:10.1007/978-94-007-5699-1_31 [35].

(vii) H. Elshaafi and D. Botvich, "Trustworthiness Inference of Multi-tenant Component Services in Service Compositions," *FTRA Journal of Convergence (JoC)*, 4(1):31-37, Mar 2013 [4] (reviewed and extended version of FTRA CSA conference paper above).

(viii) H. Elshaafi and D. Botvich, "Optimisation Based Collaborative Determination of Component Trustworthiness in Service Compositions," *Wiley Journal of Security and Communication Networks*, 2014, doi:10.1002/sec.985 (currently published online before inclusion in an upcoming issue) [5].

(ix) H. Elshaafi, J. McGibney and D. Botvich, "Attack Surface Based Security Metric Framework for Service Selection and Composition," *submitted to Inderscience Journal of Autonomous and Adaptive Systems*, 2014, (to appear) [3].

Chapter 3

State of the Art and Literature Review

3.1 Introduction

This chapter investigates current and recent research work and the state of the art in areas related to this work. It begins by discussing the different views on meaning of trust and recent convergence towards a more common understanding of the concept (Subsection 3.2.1). The application of trust to the area of service composition requires the knowledge of existing trust and reputation systems in the broader context of distributed systems. Therefore, we discuss some of the recent advances in those systems including their contributions and shortcomings (Subsection 3.2.2). We also describe threats against the systems and relate them to the characteristics of the service environments (Subsection 3.2.3).

The chapter then investigates the related work in the service environments providing justifications for the use of trust in such environments as well as discussing existing models and approaches (Section 3.3). Sections 3.4 and 3.5 discuss the recent work on topics that are required to build trustworthy composite services including business process security attributes, security metrics and QoS. The discussions identify the relevance of the existing work and its strengths. However, we particularly focus on

analysing the weaknesses that are addressed by our research in this thesis. The discussion of resource management and profitability in service environments in Section 3.6 aims to form the basis for our goal of ensuring that trustworthy compositions are also cost efficient and profitable.

3.2 Trust, Trustworthiness and Reputation

3.2.1 Views on the Concepts

In recent years there has been large amount of activity in the research area of computational trust and reputation, with applications in fields such as security [e.g. 8, 18, 36], multi-agent systems [e.g. 37–40], game theory [e.g. 41], social networking [e.g. 42, 43], e-commerce [e.g. 44, 45], and spam filtering [e.g. 46]. The terms trust and trust models are also used in Web service standards (e.g. WS-Trust [47]) but they are limited to the context of being able to trust the identity of the service [8]. However, establishing a service's identity does not mean that the service itself is necessarily trustworthy. For instance, such an authenticated service could be temporarily unreliable or unavailable.

In the field of computer security, the word *trust* is often synonymous with *security*. An example of this is the *Trusted Computing* concept [36] and its implementation by Microsoft called *Palladium* [48]. Trusted Computing focuses on securing hardware and software of a system through encryption techniques. A related initiative to the Trusted Computing called the *Trustworthy Computing (TwC)* initiative by Microsoft follows a broader multidimensional view of trust and trustworthiness. A Microsoft TwC white paper [49] states that trust is a broad concept. The paper sets four goals for trustworthy computing namely; *security*, *privacy*, *reliability* and *business integrity*. It also describes a set of methods towards achieving these goals.

Existing initiatives on trustworthy software that limit their investigation to single or narrow set of attributes including Microsoft's Trustworthy Computing are criticised by

researchers such as Hasselbring and Reussner [10]. According to the authors, the initiatives mainly focus on security while trustworthiness depends on many other attributes such as correctness, safety, QoS, security and privacy. Isolated analysis of individual attributes cannot capture the complexity in design and runtime problems, requiring a holistic approach to software trustworthiness that utilises multidimensional techniques.

In business and social contexts, the words security and trust have different meanings. Security mechanisms such as encryption, authentication, and authorization are necessary steps in establishing trust. For example, authentication mechanisms assure the service consumer that the provider of the service is who it claims it is. Similarly, a service might interact with its consumer through encrypted communication channels ensuring confidentiality. However, these security mechanisms are not sufficient as they do not assure the behaviour of the service. Instead, they only partly ensure the trustworthiness of the service. The service may not behave in the way it is required or expected in terms of reliability, availability, privacy, etc. The multidimensional view of trust has been recognised as a need or as an approach in many research studies [e.g. 7–13]. This view of trust is sometimes called *reputation-based trust* [e.g. 7, 16]. Nonetheless, reputation differs from trust in that it is more concerned with consumer and community perception of an entity, for example, based on consumer satisfaction ratings.

Initial trust definitions were limited to certain aspects of trust such as the definition by Gambetta [14] which is focused on reliability of an agent. Grandison and Sloman [15] define trust and classify trust into five classes; *provision*, *access*, *delegation*, *identity* and *context trust*. Jøsang et al. [11] defines provision trust as *"the relying party's trust in a service or resource provider, and is relevant when the relying party is a user seeking protection from malicious or unreliable service providers"*. Access trust is *"the trust in principals for the purpose of accessing resources owned by or under the responsibility of the relying party"*. Delegation trust describes *"trust in an agent that makes decisions on behalf of the relying party"*. Identity trust is *"the belief that an agent identity is as claimed"*. Context trust is about ensuring that the necessary conditions are in place. Jøsang et al. state that provision trust must be based on identity trust. Similar classifications are also suggested by other authors [e.g. 13, 16].

RFC4949 (Internet security glossary) [17] defines trust and related concepts (e.g. trust level). However, the RFC definition does not explicitly mention some key aspects such as context and relationship. Some researchers such as Chang et al. [12] and Artz and Gil [13] seek to improve the definition of trust by identifying shortcomings of definitions by previous authors.

There are therefore wide views on the meaning and scope of trust between researchers. However, we believe that trust is influenced by multiple factors and the role of those factors depends on context. This position is followed by several researchers in recent years as described above.

3.2.2 Trust and Reputation Systems and Models

Several systems and models have been developed for trust and reputation and their management in a variety of distributed environments.

REGRET [37] relies on direct experience, witness, and social information to decide on the trustworthiness (reputation) of an interacting agent in a multi-agent system while also taking context into consideration. REGRET describes two dimensions of reputation. First, individual reputation, based on an agent's own experience with another agent. This reputation value is calculated as a weighted mean of an agent's ratings giving more weight to more recent ratings. Second dimension is social reputation, which is derived from public information about an agent. It includes the reputation of the agent's group and reputation information from the trusting agent's group regarding the other agent and its group. The REGRET system requires a minimum number of interactions to reliably evaluate the reputation of an agent. It also does not describe how to build the social network on which its model depends. Additionally, it does not consider the problem of dishonest ratings.

FIRE model [38] combines multiple trust sources namely, *direct experience*, *witness information*, *role-based rules*, and *references*. References can be produced by agents that have previously interacted with the target agent certifying its behaviour.

Other factors considered in the trust calculation include context, credibility and recency of information. FIRE integrates the four sources to provide trust metrics in a variety of situations including where there is scarcity of information about an interacting agent and to enhance the precision of the trust model. However, the model lacks mechanisms to deal with dishonest or inaccurate agents providing false witness or reference information.

PeerTrust [39] is a system that uses reputation to evaluate the trustworthiness of peers. The system includes an adaptive trust model for quantifying and comparing the trustworthiness of peers based on the reputation formed through transaction feedback, and a decentralised implementation of such a model over a structured P2P network. In PeerTrust, peers use a personalised similarity measure to give more weight to opinions of peers who have provided similar ratings for a common set of past providers. However, in a dynamic large P2P system, finding such a set of partners may be difficult. As a result peers may have to select between peers for which there's no information. PeerTrust also takes a binary approach in relation to feedback i.e. satisfactory/unsatisfactory transaction. This limits the richness of feedback and the flexibility of the trust contexts.

Other popular trust and reputation models and systems in distributed environments include the following:

- *SPORAS* [50]: a centralised reputation system that extends the online reputation models such as those used in eBay and Amazon by introducing a new method for rating aggregation after each transaction without storing individual ratings. This aggregation method allows newer ratings to have more effect on the reputation.

- *EigenTrust* [51]: a reputation model for P2P systems that aims to estimate a global view of the reputation of each peer across the network.

- *PowerTrust* [52]: a decentralised system that is resistant to malicious behaviour. It requires a structured overlay DHT (Distributed Hash Table) P2P architecture, and the algorithms are dependent on this architecture. In DHT, a hash function

is used to map keys such as file names into points in a logical coordinate space. The coordinate space is partitioned dynamically among the peers such that every peer covers a region in the space.

- *TRAVOS* [53]: uses two information sources to assess the trustworthiness of peers; direct experience and witness observations. The model relies on direct experiences and only considers others' opinions when direct experience is insufficient. For this purpose, it provides a confidence metric to determine whether the personal experience is sufficient to determine a peer's trustworthiness.

Frameworks that classify and compare trust and reputation systems are described by a number of researchers [e.g. 54–56]. Understanding such frameworks is essential in developing models and algorithms for trust aggregation and prediction. Noorian and Ulieru [54] study aspects of those systems such as witness location models, reputation calculation approaches, information sources, context diversity, honesty assessment and adaptability. They then compare some of the well-known systems based on the framework features and their level of support. Aspects of the systems are classified in two categories; *soft features* and *hard features*. Attributes that help to enhance the performance of the system and quality of outcomes are considered soft features. Some of the dimensions of soft features such as reliability and honesty evaluations are developed to protect from threats in open dynamic environments such as collusions. On the other hand, hard features are characteristics that the authors identify as fundamental to building trust and reputation systems such as rating approaches, trust computation techniques and information sources. The authors highlight that when designing a trust system, the context constraints and requirements should be identified before considering what features are suitable for the system e.g. centralised vs. decentralised approach. Khalid et al. [55] discuss the components that are required to build trust and reputation models and describe phases of the trust computation process. The framework described by Hoffman et al. [56] pays more focus to attacks against those systems as discussed in Subsection 3.2.3.

3.2.3 Threats against Trust and Reputation Systems

Any usable method for determining the trustworthiness of services has to protect from threats. Such a method needs to be robust and aware of current threats that exist as a result of vulnerabilities in the trustworthiness mechanisms. There have been research activities specific to the area of threats to trust and reputation systems. Additionally, some of the threats against trust and reputation systems may not be malicious such as biased service raters.

Forging multiple identities in peer-to-peer networks in order to gain influence on reputation systems was first described by Douceur [57] . This form of attack was named *Sybil attack*. A variety of other attacks as well as defence mechanisms in reputation systems are described by Hoffman et al. [56]. They discuss a framework that analyses the structure of existing reputation systems and attempts to classify attacks against reputation systems by identifying which system components and design choices are the target of each category of attacks. Examples of those components and choices are the types of input and output metrics, redundancy and distributed architecture of a reputation system. The authors also describe how to integrate the defence mechanisms into reputation systems to become resilient to attacks. The categories of attacks described by the authors include the following:

- *Self-Promoting* node maliciously increases its reputation.
- *Whitewashing* exploits a system vulnerability to improve bad reputation (similar to Sybil attacks).
- *Slandering* by providing false reports.
- *Collusion* between attackers.
- *Denial of Service* to block the computation and distribution of reputation updates.

The work of Hoffman et al. provides a useful analysis of aspects of reputation systems that can pose vulnerabilities to attacks from malicious users and providers. The authors don't discuss some possible types of attacks that are more specific to the service environments. Examples of such attacks are:

- *Free-riding*: a component service unjustifiably benefits from the high trustworthiness ratings of a composite service, and
- *Camouflaging*: low trustworthiness of a component is hidden through insertion into a trustworthy CS.

Additionally, prediction of service trustworthiness and distribution of existing attribute values from composite to component services may result in unfair values if the techniques don't recognise certain factors in the computations. Examples of such factors are the variation of importance of component services in a composite service and the structure of the service execution process.

In addition to their discussion of types of attacks against trust and reputation systems, Jøsang and Golbeck [58] describe the need for robustness evaluation of such systems in practical environments. The purpose is to evaluate which attacks are realistic, for example due to the existence of incentives for those attacks.

Another type of attack which is described by Mármol and Pérez [59], is Man in the Middle attack where a malicious peer may intercept messages from a benevolent service provider peer to a service requester and rewrite them with bad services. Consequently, the attack results in the reduction of the reputation of the benevolent peer. That participant could also maliciously modify the recommendations given by an honest peer, in order to benefit own interests. This type of attack can be overcome through encryption of messages and communication channels.

Research on addressing dishonest raters problem include the use of *majority rule* such as the methods discussed by Walsh and Sirer [60] and by Malik and Bouguettaya [61]. According to the majority rule, ratings that are far away from the majority's

opinion are considered unfair ratings. The majority-based methods can be ineffective in highly malicious environments; for example, where dishonest raters are more numerous than honest raters. Another approach by Yang et al. [62] describes a detection mechanism of colluding dishonest raters using signal modelling where a detector considers honest ratings as noise and unfair ratings as signals. A set of detectors are set up to identify different patterns of attacks. However, the dependence on the patterns of ratings can make this approach highly susceptible to false positives and false negatives.

3.3 Trustworthiness of Composite Services

3.3.1 The Need for Trustworthiness in Service Compositions

In this subsection we discuss some of the important reasons and justifications described by researchers for the consideration of trust in the composition of services. It also describes the basis of some of the techniques developed in thesis such as the collaborative determination of component trustworthiness. The challenges that face the related research in the service environments are also discussed.

Takabi et al. [9] discuss barriers and possible solutions to providing trustworthy services within cloud computing environments. Although their discussion focuses on cloud computing, many of the concerns and the suggested approaches to solving existing and anticipated problems are also applicable to service architectures in general. They describe the need of multiple service providers to collaborate and compose value-added enterprise services. They propose that a trust framework should be developed to allow to efficiently capture parameters required for establishing trust and to manage evolving trust and shared requirements. Furthermore, customer behaviour can evolve rapidly, thereby affecting established trust. They, therefore, describe the need for a framework that helps establish, negotiate, and maintain trust.

Takabi et al. also highlight other relevant issues investigated in this thesis including QoS and pricing as critical in service search and composition. They state that these issues must be addressed to describe services and introduce their features. These aspects also can support finding the best interoperable options, integrating services without violating their policies, and ensuring that *Service Level Agreements (SLAs)* are satisfied.

The need to establish trust between Web service consumers and providers is required according to Malik and Bouguettaya [61] as a result of a specific characteristic of the service environments. The dynamism of the environment allows the existence of consumers and providers that are unknown to each other. Since the services would be mutually unfamiliar, the decision to engage in a transaction with another Web service would not be based on any prior direct experience. This makes it challenging to establish a priori, whether and to what extent a given service may be trusted to carry out the required functionality.

Some other challenging research areas related to this PhD study such as QoS aware composition, business driven composition, and composition of resources are described by Papazoglou et al. [63]. They state that services collaborate in highly distributed environments, naturally cutting across organizational boundaries. These environments require that contracts are set up, specifying agreements between services regarding their collaboration, both at functional and non-functional levels. These contracts may serve as the basis for process monitoring and adaptation. Such agreements can be used in the collaborative determination of component trustworthiness as discussed in Chapter 6. Some of the current research activities according to the authors are focusing on providing facilities to dynamically check the state of an agreement.

Existing approaches according to Singh [64], fail to adequately address the challenges for trust in service oriented computing. He states that, for Web services to be effectively composed they need to be trustworthy and to be trusted by their users as well as other collaborating services. More recent research considers aspects such as dynamics and adaptation of trust. For example, Skopik [65] regards the introduction of the concept of a dynamic and adapting trust as an intuitive approach to addressing the

challenge of open loosely coupled service environments. The author claims that taking into consideration trust relations when selecting services and other resources leads to more efficient cooperation and compositions of services. Skopik states that trust and reputation mechanisms are key to the success of open dynamic service-oriented environments.

In dynamic service environments, services may change behaviour resulting in variation in the level of trustworthiness over time. Additionally, operational or business-related requirements may result in changes in expectations. A more trustworthy atomic service may emerge that could create an opportunity for composite service providers to enhance their service trustworthiness. Therefore, trustworthiness techniques need to adjust to the changing circumstances and update trustworthiness in a timely manner. In addition, those techniques should not only determine service trustworthiness but also support the maintenance of the required level of trustworthiness.

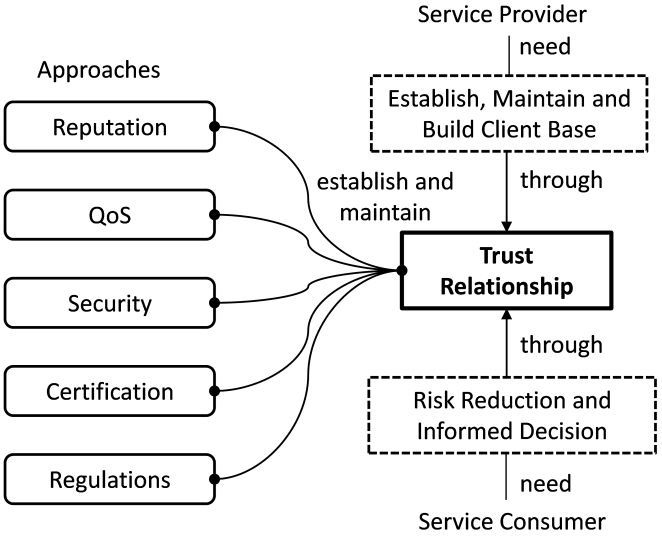


Figure 3.1: Categories of service attributes that can be taken into consideration in calculating service trustworthiness and the role of trust in business (modified from Malik and Bouguettaya [61])

Figure 3.1 shows the main approaches commonly used in establishing and maintaining trust by consumers towards services and service providers. The service provider aims to build and expand its client base through maintaining its service's trustworthiness. The consumer benefits from the trust by reducing risk and making more informed

decision in relation to the service use. The figure is modified from Malik and Bouguettaya [61] which included self-regulation as one of the traditional approaches in building trust. However, the authors state that self-regulation has not been effective in establishing trust and it may not be possible to validate provider's claims prior to transactions. Third party certification in the view of the author of the thesis is a reference that helps assure provider's trustworthiness for consumers rather than an experienced or monitored attribute of a service. Therefore, we consider certification outside the scope of this thesis. In the thesis we add QoS as one of the approaches as it clearly affects trustworthiness as discussed in previous sections. Malik and Bouguettaya appear to mix between the concepts of trust, reputation and QoS (they refer to it as QoWS) and often use these terms interchangeably. However, we consider QoS as objective evaluations of the status of a service from a number of aspects called trustworthiness attributes such as reliability, availability and response time as opposed to their subjective QoS evaluation.

3.3.2 Trust and Reputation Systems in Service Environments

The subsection explains lessons learned from the research and developments in trust and reputation for atomic services, service composition and service oriented computing in general particularly in relation to their strengths and weaknesses.

3.3.2.1 Trustworthiness of Individual Services and Providers

A framework for establishing trust in service oriented environments named *RATEWeb* is developed by Malik and Bouguettaya [7, 61]. The framework operates by aggregating reputation ratings from consumers in a P2P fashion. The different ratings are aggregated to derive a service provider's reputation. The concept of reputation is used interchangeably with QoS (called QoWS) in RATEWeb. This in turn is used to evaluate trust. RATEWeb aims to support trust-based selection and composition of Web services. The framework uses the concept of communities to cluster Web services based on domains of interest. Communities are also used to bootstrap the reputation of newcomers and setting policies relating to members when it decreases below certain

levels. At the end of an interaction a service consumer rates the provider according to predetermined criteria. These ratings are then used to update the provider's reputation. RATEWeb also deals with issues relating to rating recency (temporal sensitivity), personalised preferences and rater credibility. Rater credibility is called feedback trust as opposed to service trust.

RATEWeb provides a variety of mechanisms for dealing with a number of problems that face reputation systems as described above. However, it has a number of drawbacks. One of the drawbacks is that it is not a readily implementable model. The authors do not discuss how it can be implemented in real-world Web services environment or how it interoperates with Web service standards and current practices. Another problem is that the framework evaluates the reputation of service providers (possibly offering multiple services) rather that their services. It also does not provide mechanisms for responding to runtime changes in the environment that may affect a provider's trustworthiness. Reputation mechanisms that support service composition (e.g. aggregation of component service reputations) are not detailed enough. The work does not take into consideration composition structure and component importance differences in the computation of trustworthiness.

While Malik and Bouguettaya investigate community clustered services, Skopik et al. [65, 66] discuss trust mechanisms in *mixed service-oriented systems*. They describe those systems as open ecosystems made up of both human- and software-based services. In the trust model described by Skopik et al. the metrics used in determining trust depend on the context and the environment. Examples of such metrics are the actor responsiveness (e.g., measured by an average response time), the reliability in responding to requests, and the ratio of performed to delegated tasks. The applicable environment described by the authors appears to have characteristics from both social networks and service environments which makes the domain for application of the approach very large, abstract and possibly difficult to customise to suit a particular environment. Another challenging aspect of their work is that it assumes access to the content and the properties of exchanged messages between actors including human-human, human-service, and service-service interactions. This clearly raises security

and privacy concerns.

Some approaches proposed by researchers to establish trust fully depend on design time verification processes. For example, the use of formal specification to assure the behaviour of services is described by Dragoni [67]. This approach for trust is based on service contracts called *Trust-by-Contract*. A contract is defined as a formal specification of the trust behaviour of a service that a software developer binds and publishes with the service. The author claims that existing social trust models have inherent weaknesses in the services environments such as the dependence on belonging to communities. However, complete reliance on code verification in trusting services ignores runtime behaviour such as reliability and availability. Such approaches therefore do not ensure full trustworthiness of services.

Other researchers such as Pranata et al. [68] develop their trust establishment approaches with protection from known threats in mind. The mechanism presented by Pranata et al. aims to determine trustworthiness of service providers. The mechanism, which utilises the Web of Trust (WoT), aims to improve the precision of trust computation by considering raters' feedback, number of transactions and credibility in determining a provider's trustworthiness. The authors also consider other improvements such as incentivising raters' participation through a *willingness to rate* parameter, raters' categories, and safeguarding against changing identities. However, the specific categorisation of raters into *trust circle*, *recommender* and *unknown raters* to protect from malicious raters limits the usefulness of the proposed mechanism in service environments that require more flexible solutions.

Several works determine trustworthiness completely based on reputation and subjective consumer ratings. Spanoudakis and LoPresti [69] propose a model of runtime trust assessment of services that aims to support trust based on monitoring evidence together with subjective assessments from service consumers. A framework proposed by Majithia et al. [70] uses reputation in service discovery. In this framework, service consumers submit their ratings to a *Reputation Manager Service* which computes the service's reputation based on those ratings. Maximilien and Singh [71] present an

agent based trust model for service reputation that enables rating of individual services as well as providers. The model has shortfalls, such as the need for human intervention.

3.3.2.2 Trust in Component Service Selection and Composition

Several researchers such as Paradesi et al. [72], Hang and Singh [73] as well as Li and Wang [74], investigate the area of trust in service composition and selection. They study issues related to deriving the trustworthiness of compositions from that of component services and the selection of the most trustworthy components. For example, Li and Wang [74] describe an algorithm for trust evaluation in composite services that takes into account component service invocations based on the service workflow. Although their approaches are useful in solving certain aspects of research problems in computing composition trustworthiness, they do not investigate the determination and evaluation of component trustworthiness attributes.

Some of the existing works do not consider composition structure and other characteristics of business processes in the evaluation of trustworthiness. For instance, Mehdi et al. [75] model the trustworthiness of atomic or composite services using probabilistic models based on the distribution of counts of quality classes of their previous interactions. The authors defer the consideration of the structure of composite service for future work. They also rely on direct interactions with a service in order to compute trustworthiness which are not often available as discussed above.

The use of service trust updates in the maintenance of service compositions and to influence their adaptation is one of the areas that are closely related to this thesis. In our work, the determined attribute values for the component services can be utilised by CSPs to improve trustworthiness of their composite services. This can be accomplished through service selection and adaptation by replacing untrustworthy components. Ciszkowski et al. [76] propose a framework that enables providers to facilitate composite service adaptation according to consumer expectations and maintain QoE at a satisfactory level.

Collaborative determination of component trustworthiness depends on joint invocation of CS components. It also requires sharing of information between CSPs. Arenas et al. [77] describe a reputation management scheme for collaborative systems where organisations share resources such as services to form virtual organisations. The model is an example of collaborative scenarios in existing service environments where our approaches can be useful.

The relationship between service selection and resource management is discussed by Abawajy [78]. The author describes a framework for determining the trustworthiness of federated cloud computing entities. This framework aims to support the selection of trustworthy clouds to peer with and outsource applications for execution or data storage. Further details of this work are discussed in Subsection 3.6.2.

3.3.3 Propagation of Updates from Composite to Component Services

When a composite service is provided to consumers, their ratings will reflect their experience and perception of the service as a whole. However, component services may have different trustworthiness. Additionally, the component contributions to the trustworthiness of the composite service may also be different based on their role in the composition. Therefore, a method is required to accurately and fairly distribute the reputation updates to the component services. Any usable method for determining the trustworthiness of component services based on their CSs has to strive to meet certain requirements. Some important requirements include accuracy, fairness and protection from threats. Such a method needs to be robust and aware of current threats that exist as a result of vulnerabilities in the trustworthiness determination mechanisms especially when monitoring of the behaviour of individual components is not possible.

Despite the abundance and variety of research in the area of trust and trustworthiness, there is no existing work, to our knowledge, addressing the collaborative determination of trustworthiness attributes of joint component services based on the attributes

of their compositions. Our approach detailed in Chapter 6, which addresses the collaborative determination using optimisation, has advantages over existing works that aim to determine trustworthiness attributes of components from the corresponding attributes of their compositions. That is because existing work such as that from Nepal et al. [79] only considers the determination of component attributes from an individual composite service.

Other works, such as those by Malik and Bouguettaya [61], Paradesi et al. [72] and Hang and Singh [73] discuss how to aggregate component attributes and/or how to select component services for a new composition. Those works do not consider how the trustworthiness attributes of the component are evaluated or determined in the first place other than through direct interaction of the component with consumers. However, component services may not always have direct business value as atomic services unless used and evaluated within a composition of services. Therefore, our work proposes a solution to an important problem that has not been addressed in the current literature. Other advantages of our work are also highlighted in the next paragraphs.

Nepal et al. [79] present an approach that aims to fairly propagate reputation values from a composite service to its components. However, unlike our work, they only determine those values using individual composite services. The authors do not consider a distributed collaborative approach or even benefiting from the existence of multiple composite services that invoke the same component. Their approach propagates the reputation values based on component contributions to the composition where these contributions are assumed to be proportionate to the historical reputation values of components. According to the authors, reputation could be a single value representing an overall perception or a vector representing a value for each QoS attribute such as performance, reliability and availability. They claim that a component service that has higher reputation is likely to have a higher contribution towards the overall reputation of the composite service as perceived by a service consumer than components with lower reputation. However, this is often not true because a component can have high reputation but still less important to achieving the goal of its composite service. For example,

an insurance service (see scenario in Chapter 2) may have a high reputation in an e-commerce composite service due to its own performance but it is not as important as the buying and shipping components in meeting expectations of the service consumers.

Additionally, the reputation of a CS is computed by Nepal et al. as the average of that of its component services. This can cause inaccuracies because the resulting reputation value can hide untrustworthy components and cause bad consumer ratings for seemingly high reputation CSs. For example, a single failing component can cause the unreliability of a multi-component service. Another relevant example is discussed by Meland [80] where a component may affect the trustworthiness of a composite service through what is called *service injection* where a malicious component is injected into dynamic service compositions.

Other attempts to solve the problem of distributing the feedback scores of composite services to their components include the work by Wen et al. [81] who distribute the scores based on the importance of each component in the contribution to the service success and failure.

Since service reliability is one of the important trustworthiness attributes, Chapter 6 discusses the correlation between the reliabilities of multiple existing composite services to determine the reliabilities of their components. The approach is also usable to optimise the selection from candidate components of new composite services. One of the existing research that is most relevant to our work in this area is that by Zheng and Lyu [82]. The authors propose a reliability prediction approach for services to exploit the past failure data of similar users through collaboration in order to predict service failure probabilities for a current user. The predicted failure probabilities of the atomic services are composed using different compositional structures to predict the reliability of a composite service. However, they consider exploring failure correlation between different services as a future work.

3.4 Security Attributes in Trustworthiness

3.4.1 Security Attributes of Atomic and Composite Services

Information security research activities in the areas of service oriented computing and business processes [e.g. 83–86] examine a wide range of associated security problems and propose solutions. These research activities are helpful in order for the thesis to investigate the application of security metrics to services and business processes. For example, Bertino et al. [83] propose an authorization model called RBAC-WS-BPEL for business processes that supports the specification of access constraints. Such a model can limit the extension of the attack surface of composite services as we discuss in Section 5.4.

Research on the integration and enforcement of security features into business processes is one of the currently active research areas. A BPMN based metamodel that allows the integration of security requirements into BPMN models is presented by Rodríguez et al. [85]. The metamodels can then be refined by security experts and implemented by developers with a view to support an MDA (Model Driven Architecture) approach to convert the security annotated model to an executable model.

A security attribute or property is defined in the *Aniketos* project [87] as "a characterization, attribute or specification of a service that realises a security requirement or policy. E.g. service A offers encrypted communication using a 1024-bit RSA key, only stores data locally for 24 hours or it only allows one instance of an identity to be logged in concurrently". In order for a composite service to be fully trustworthy it has to be verifiably secure. Verification of the security attributes of a service can be based on various mechanisms depending on the attribute considered. It includes manual verification and automated verification. Automated verification includes formal (mathematical) or conventional non-formal verification techniques. For example, static code analysis tools can be used to verify against certain vulnerabilities such as buffer overflows.

Verification techniques are necessary but may not be enough to verify the compliance of a service to the requirement of a security attribute. This can be attributed to the fact that the non-existence of a vulnerability may not mean that other related types of vulnerabilities exist that can result in a breach of a security attribute. Additionally, the code-based verification techniques may not always be perfectly accurate. Finally, the verifiability also depends on the accurate definition and the granularity of the security attribute itself. OWASP Code Review Guide [88] states that code review is one of the layers in a defense-in-depth approach to application security. Therefore, monitoring of security attributes compliance is important as part of the enforcement strategy of those attributes.

Dependence on manual annotations and deliberate exclusion of certain aspects of the code are some of the reasons for the difficulty in producing software that is free from vulnerabilities as explained by Shahriar and Zulkernine [89]. The authors present a taxonomy and classification of the existing approaches for monitoring software vulnerability exploitations and attacks. They classify the approaches based on a set of characteristics e.g. implementation mechanism. Then, they present a taxonomy by classifying the approaches based on monitoring aspects that distinguish those approaches. The authors state that guaranteeing absence of vulnerabilities during development (using methods such as static analysis and vulnerability scanning) is extremely difficult. Therefore, complementary runtime monitoring techniques to detect application vulnerability exploitations are needed.

Access control can be a specified security attribute of a composite service or possibly a mechanism for compliance with requirements for certain security attributes. For example, Separation of Duty (SoD) can be a security property of a composite service that is accomplished through role or attribute based access control. Brucker et al. [84] describe a tool chain that aims to support the design-time modelling and runtime enforcement of security requirements for business processes. They use both access control and SoD as security requirements that can be modelled in a BPMN business process. Their security annotated BPMN design-time modelling is an extension of *Activiti Designer* [27] and for the enforcement of access control, XACML policies and XACML

compliant PEPs (Policy Enforcement Points) are automatically generated.

Dynamic and flexible trust systems that incorporate reputation with other criteria for access control can support policy based decisions at runtime. Such scenario also requires runtime monitoring and notification systems. Blaze et al. [90] discuss dynamic trust management in service environments based on their previous works related to distributed systems [18]. Existing trust management approaches, according to the authors, are insufficient for SOA. The main problem is that policies specified by existing trust management systems are static. However, in order to support SOA, the authors state that a policy specification should be dynamic to accommodate changes in both the system and its environment. For example, dynamic change in alternative services and their availability need to be accommodated in the trust management system knowing that the offered and required security and trust features may vary among services as well as among requesters. The authors claim that at the time of their work, there was no mechanism through which a security policy could re-evaluate the user or service privileges and take some necessary action e.g. modifying authorizations. One of the objectives of our work is to support runtime monitoring and update of trustworthiness and allow CSPs to use this feature to make decisions based on the dynamic trustworthiness changes of the components.

Security attributes of services can represent an initial step in establishing trust where runtime data may not be available yet for the services and their components. Khan and Han [91] propose using logic programming to represent security attributes of atomic software components and reason about their compositional matching with other components to build trustworthy component-based software. According to the authors, software composers often have to compose systems with components for which they have partial or no knowledge about their underlying security attributes. However, the paper assumes the existence of a global certifying authority that approves components with their exposed security attributes that the component claims. The certificate ensures the matching between the actual implementation and the exposed security attributes. As we describe in the research scope (Section 2.3), certification can serve as evidence of

security attributes. This can be part of attack surface modelling and evaluation of components (Subsection 5.3.3).

3.4.2 Modelling and Quantifying Security Attributes

Security is an important aspect of trustworthiness of any service. However, security is qualitative and difficult to measure. The area of security metrics has received significant attention from the security research community. One of the main challenges for the research in this thesis is how security is quantified in order to integrate it with the remaining trustworthiness attributes.

Approaches to modelling of software security attributes, vulnerabilities and threats present an ongoing research challenge in the security community. The approaches may depend on the purpose of the model such as identifying threats, their potential damage and prioritisation. The models may apply in different scenarios such as network configurations and software development processes. Examples of such models are attack trees and graphs [e.g. 92–94]. A related concept is the defence (or protection) trees and graphs [95, 96] which extend attack trees by providing countermeasures and mitigation of security risks that result from identified vulnerabilities and threats.

There are also research efforts to classify security threats specific to services. Vorobiev and Han [97] provide a classification of security threats in Web services in order to support distributed collaborative defence mechanisms for Web services.

Another related area to attack graphs is *vulnerability metrics* which aim to assess risk based on severity of system vulnerabilities. For example, *Common Vulnerability Scoring System (CVSS)* metrics [98] can be used to indicate certain security levels for a single vulnerability. However, risk may also depend on other criteria such as the location of vulnerability within the network. For example, a quantitative model may aggregate metrics that measure the likelihood that breaches occur within a given system configuration, taking into consideration the effects of context and possible dependency between vulnerabilities.

The security metrics field is seen as an important factor in decision making in various areas related to security. Examples of such areas are discussed by Jansen [99] and by Bau and Mitchell [100]. These include the design of security controls, determining the robustness and efficiency of security operations as well as comparative evaluations particularly for emerging computing paradigms such as service-oriented and cloud computing as.

The concept of attack surfaces [101, 102] is gaining more attention in the security research community. The concept is particularly relevant to the areas of security testing, updates and configuration of software. It entails identifying entry points and exit points in a software system which are then used to determine the relevant potentially exploitable system resources. Howard et al. [101] introduce the notion of attack surface and propose a measurement method for the attack surface of the Windows operating system. The method assigns weights to Windows attack vectors to reflect their likelihood of being used in attacks. Those weights are the contributions of the attack vectors to the attack surface. Manadhata and Wing [102] extend the work of Howard et al. by formalising the approach, providing more systematic method and widening its application to a limited extent to other software systems particularly systems developed in the C programming language. They classify system resources into methods, channels and data items. Their work does not take into consideration some aspects of security that influence the attack effort and damage potential and consequently an attack surface as a whole. Additionally, the authors regard the composition of attack surfaces as a future work.

A systems' assurance model named Structured Assurance Case Model was developed originally by Kelly [31]. The model has been adopted widely in assuring system safety while some researchers such as Rhodes et al. [32] investigate its applicability to software. Rhodes et al. state that assurance case models currently do not address the assurance of systems of systems (e.g. service compositions). Boland et al. [103] propose an approach using SACM for the determination of software trustworthiness. Weinstock et al. [104] describe an approach based on SACM that aims to discover

vulnerabilities caused by code defects with the buffer overflows as an example. The authors attempt to develop a pattern for a more general implementation of their approach. Ouedraogo et al. [105] describe a set of metrics to help evaluate the security assurance of runtime systems. Such runtime evaluations can be beneficial to our framework in augmenting static evidences in the structured assurance case through including runtime and operational evidences.

In our work, we utilise the SACM model in measuring security attributes based on documented design-time or runtime evidences in order to assess the security aspects and consequently the attack surface of component and composite service resources. There are models that can be considered as a basis for security metrics such as Microsoft STRIDE [106]. The focus of these models is on identifying threats from an attacker or vulnerability perspective. In fact, Shostack [107, 108] at Microsoft recognises that threat modelling based on his experience is disconnected from the software development lifecycle. Nonetheless, detection of threats and vulnerabilities can serve as an extension to our approach. Therefore, we consider it an interesting and significant future work as we describe in Chapter 8.

Our work focuses on internal characteristics of services. It needs to evaluate exploitability of resources belonging to a service and their defence methods regardless of specific threats or known vulnerabilities. This is performed through evaluating sensitivity, effort and damage that can be caused in accessing a resource. Therefore, models such as STRIDE are not suitable for this purpose. The process provided by SACM through claim, argument and evidence together with the attack surface concept meet the requirements of our approach. The model facilitates a structured process in measuring security that uses material evidence based on reported evaluation of attributes of resources. Using arguments the model assesses the evidence support for a claim that a resource is secure in relation a particular aspect.

Another notable work on security metrics is a book by Jaquith [109]. He describes methods for quantifying security through metrics and the approaches to the development of such metrics. The author focuses on the business value of the security measures

of an organisation. He suggests that a risk management framework is needed in which organisations can quantify the likelihood of danger, estimate the extent of possible damage, understand the performance of their security organisation, and weigh the costs of security safeguards against their expected effectiveness. According to the author, metrics need to be objective, easy to collect and expressed in numbers and measurement units. The book classifies the metrics into a number of categories each with a list of metrics specifying each metric's purpose and source. Although each metric's description specifies how to measure it (e.g. percentage of systems, number of incidents), no solution is provided on how to normalise the metrics' values and provide a general view of the risk to the organisation or its current security status. The author recognizes that the identified metrics are based on observations and practices rather than a modelling perspective.

3.5 QoS in Service Compositions

3.5.1 QoS Attributes and Their Aggregation

The aggregation of component QoS attributes to estimate the QoS of composite services is required in multiple stages of the service lifecycle. It is important to know the QoS for a CS during the design phase and service discovery and selection to ensure that consumers receive the appropriate QoS level. During runtime, the aggregation of the QoS attributes is part of the monitoring process of the trustworthiness of a CS to ensure that the QoS metrics do not deteriorate or decline below, for instance, specified thresholds. The aggregation techniques are also important if alternative plans have to be taken in cases of unsatisfactory performance of a CS or because of changes in the environment or the requirements. Consequently, the techniques will support those plans for adaptation or recomposition of the CS.

Methods for the aggregation of some QoS attributes are proposed by a number of researchers [110–112]. Cardoso et al. [110] use the term *fidelity* to refer to the

properties of a good or a service; a concept somewhat analogous to reputation. In addition, they develop a model to aggregate a limited set of QoS attributes including workflow task time, reliability and cost. An algorithm is also described to compute the QoS of a workflow. The algorithm applies a series of reductions to a workflow, until only one task remains which represents the workflow QoS. The model and the algorithm are implemented as part of a workflow management system prototype called METEOR that supports workflow design and monitoring. A more detailed investigation of several workflow patterns is provided by Jaeger et al. [111]. Jaeger et al. take an approach of aggregation of QoS attributes in terms of lower and upper bounds of their values.

The aggregation of reputations (fidelities) for a composition by Hwang et al [112] using the sum of that of its components can result in trust vulnerabilities. Summing the component fidelities will clearly not reveal the existence of few components with bad reputation in the workflow, even when those components can significantly damage the reputation of the workflow as a whole. In addition, the paper does not describe the aggregation of security attributes.

Reliability of component and composite services receives added attention by researchers [e.g. 30, 82, 114]. For example, Grassi and Patella [114] present reliability prediction mechanisms that aim to satisfy the requirements for decentralization and autonomy. They propose mechanisms to recursively aggregate the reliability of a CS based on those of its components.

Several research studies exist on relating consumer satisfaction, QoE, service quality and quality dimensions. Lee and Lin [23] identify a set of service quality attributes and relate them to consumer satisfaction. Their survey indicates the main quality dimensions that affect consumer satisfaction are reliability, security and responsiveness. Li and Suomi [24] propose an eight-dimension scale to measure service quality based on the commonly used SERVQUAL scale [115]. Udo et al. [25] examine the impact of service quality on perceived satisfaction and other consumer behaviours.

3.5.2 QoS in Service Selection

The problem of QoS-based service selection, particularly of individual services, has been considered in several research works. Wang and Vassileva [116] classify QoS metrics used in Web services selection into four categories; performance, dependability, security and application specific metrics and list attributes in each category based on a W3C document on QoS requirements for web services [117].

A mechanism for dynamic Web service selection based on reliability is described by Hwang et al. [113]. They aim to determine a subset of Web services that can be invoked in order to successfully orchestrate a composite Web service. They propose using a finite state machine to model the permitted invocation sequences of Web service operations. Each state of execution is assigned an aggregated reliability to measure the probability that the given state will lead to successful execution in the context where each Web service may fail with some probability. The inclusion of other QoS metrics in the proposed approach was considered a future work.

A different approach using genetic algorithms is proposed by Jaeger and Müller [120]. They discuss the optimisation problem in selecting services when considering QoS attributes. They describe the application of a customisable genetic algorithm and compare its results and performance to the use of other heuristic approaches.

Common problems caused by relying on subjective consumer feedback include biased ratings and lack of incentives to rate. Limam and Boutaba [118] attempt to solve these problems based on the *expectancy-disconfirmation theory* from market science. They describe a framework for service selection based on quality, cost, and reputation. An objective automated rating model based on the theory is defined to overcome feedback subjectivity issues. The authors claim that consumers are often more likely to leave negative feedback when they are dissatisfied with the experienced service than to leave positive feedback when they are satisfied leading to unfair reputation reports. The expectancy-disconfirmation is used for explaining or predicting the acceptance or rejection of marketed products. The ratings are then aggregated to provide reputation value

for the service selection. However, the expectancy-disconfirmation marketing theory although widely used it has also received much criticism in relation to its validity and reliability. For example, Yüksel and Yüksel [119] state that the the use of expectations might be less meaningful for services than for tangible consumer goods that are easy to evaluate prior to purchase. They also argue that while predictive expectations are often used as the comparative approach in the theory, there is insufficient research evidence on whether consumers use only predictive expectations in their post-purchase product evaluations and that satisfaction processes may differ across products/services.

The use of *Multiple Criteria Decision Making (MCDM)* using QoS attributes for service selection is also investigated by researchers [e.g. 121, 122]. Huang et al. [121] apply MCDM with a weighted sum model to allow service requesters to evaluate services. Their method transforms the QoS-based optimization of service composition into a mathematical programming problem by deriving the objective functions of constituent workflow constructs. In our optimisation based method of collaboration between composite services to determine component trustworthiness attributes, we represent each composite service by an equality constraint and the objective function corresponds to the aggregation technique of candidate construct. Each constraint corresponds to the structure of the process of a particular composite service. See details in Chapter 6. A multi-criteria service selection method called *Exponential Weighted Difference* is proposed by Zia ur Rehman et al. [122]. The purpose of the method is to restrict the effect of mutual cancellation between criteria (attribute values) exceeding and below specified requirements. However, the article only focuses on the performance attribute of services. Lécué [123] combines the semantic similarities between output and input parameters of web services (functional criteria) together with QoS criteria to rank and optimize compositions.

3.6 Business and Operational Perspective

Other objectives of service providers may be as important such as cost efficiency and profitability. These need to be balanced with trustworthiness to ensure that practical

aspects are considered. Other relevant aspects of services are resource limitations and resource management objectives which need to coordinate with trustworthiness monitoring and prediction mechanisms.

3.6.1 Business Aspects of Services and Compositions

To our knowledge, there has been no specialised research in the area of pricing and profit optimisation of composite services other than the consideration of costs in the optimisation of composite services. The aggregation of costs of composite services was considered as a factor in several research works on component service selection and composition optimisation techniques such as Jaeger and Mühl [120], Hwang et al. [112], as well as Ardagna and Pernici [124].

Some guidelines that can be used in research on pricing and cost management of composite services are detailed in a book by Irene Ng [125]. The author discusses several aspects of revenue management including the role of capacity and supply in a service's profitability, price discrimination, and other pricing related issues. An example of the role of capacity is the *capacity-based pricing* which is often used when services are sold based on how much capacity has been used up. Hence, when there is excess capacity, the price is low and as the available capacity becomes limited, the price starts moving up too. This means that the provider can either control demand to match supply or control supply to match demand. Otherwise, there would be a loss of revenue from either unused or insufficient capacity. The author also discusses the advantage of bundling in maximising the usage of available capacity, reducing costs and increasing revenue.

Revenue management and profitability of composite services can benefit from existing mechanisms and solutions in relevant areas of research on the business management of products. Service composition is similar in some aspects to product bundling by companies serving their consumers with heterogeneous preferences. The pricing models that have developed in the area of bundling are useful in relation pricing of service compositions as we discuss in Chapter 7. Bundling has been studied in terms of

consumer behaviour, economics and marketing [e.g. 126, 127]. Bitran and Ferrer [126] address the problem of determining the components and price of a bundle to maximise the total expected profit in a competitive environment. According to the authors, bundles are sets of components that must meet technical constraints. A company's objective is to build a bundle and offer it in a market where it will compete with other bundles. Consumers purchase the bundle that maximizes their utility after examining all available bundles. The company selection of the bundle's components and its price is made based on the competing bundles and the uncertainty in the consumer choice process. Bitran and Ferrer propose a procedure to determine the optimal composition of the bundle and the price at which it should be offered. The paper includes a discussion of the cases that consider multiple customer segments and multiple bundles. A constant negative value is used to represent the *price response function* which is not realistic since demand may change differently at different price changes. *Attractiveness* is considered an attribute of a bundle in addition to price. The value of the attractiveness factor is the weighted sum of the attractiveness factors of the bundle's components. The price is not considered in the component attractiveness as it is treated separately. The authors suggest that the attractiveness factor is based on customer surveys.

The work by Chung and Rao [127] investigates market segments for bundles. They describe a model to find market segments for any type of bundles including those with heterogeneous products in multiple product categories. They aim to estimate the *willingness to pay* and to determine optimal prices for market segments.

Profitability may be achieved through dynamic and flexible mechanisms by considering objectives and constraints such as the efficient use of offered service resources, consumer satisfaction and consumer budgets. An approach to maximise profit for providers while offering resources to consumers proportional to their willingness to pay is discussed by Tsakalozos et al. [128]. The authors aim to solve the problem of managing at runtime the efficient use of shared service resources while maintaining satisfaction of consumers who have limited budgets (value for money). Their approach requires automated monitoring of response times and adjusting the amount of resources

accordingly. The work however, does not consider conditions in which providers require composition of services or resources for their consumers. The consideration of admission control together with pricing is proposed by Marbukh and Mills [129]. They describe a model for optimisation of pricing and scheduling in order to maximize provider revenue. The authors suggest that combination of pricing on the longer term and admission control on the shorter term are required for revenue maximization. However, the model assumes that utility of service consumers is proportionate to the response time rather than trustworthiness. As we describe in our work trustworthiness encompasses response time as well as other important factors in consumer satisfaction.

Another area related to characteristics of composite services is a category of products supply approach called *flexible products* [130]. Flexible products allow the substitution by a supplier between alternatives and can therefore help to maximize overall revenue as well as capacity utilization in markets with highly uncertain demand. Flexible products have the advantage of increasing overall demand and improving capacity utilization. The research in this area is relevant to the capacity management of composite services especially dynamic composite services to maintain trustworthy and profitable services. Petrick et al. [131] describe mathematical models that have been developed for capacity control of flexible products that can be used to exploit the substitution opportunities together with their strengths and weaknesses. Applying these concepts to the composite services, they illustrate the importance of such dynamic control mechanisms in maintaining the trustworthiness of composite services.

3.6.2 Resource Control Systems and Mechanisms

The relationship between trustworthy service selection and resource management is discussed by Abawajy [78]. The author describes a framework for determining the trustworthiness of federated cloud computing entities. This framework aims to support the selection of trustworthy clouds to peer with and outsource applications for execution or data storage. He states the open and dynamic nature of these environments as well as the independent capacity planning and provisioning of resources to consumers

make resource sharing a challenging task. Therefore, there is a need for mechanisms for these clouds to peer with each other and for admission control when accepting requests originated from other clouds. The framework proposed aims to support the selection of trustworthy clouds to peer with and outsource applications for execution or data storage. The work of Abawajy highlights the need for trustworthiness-related collaborative resource control which is one of the areas studied in this thesis.

There has been active research in recent years on resource allocation for services including compositions [e.g. 128, 129, 132–134]. Research often considers resource allocation with some related aspects considered in our thesis such as profitability, admission control, consumer differentiation and QoS. Wu et al. [132] describe an approach to improve the throughput of a static CS workflow with a common resource pool. The authors seek to improve resource allocation through tracing and prediction of workload dynamics of component services as requests traverse through the workflow. Their work takes into consideration several factors that can affect workload such as service execution time, transition probability, replication and setup times for additional capacity and the uncertainty in request arrival time. Their overall goal is to maximise the throughput of workflows within available resources. The work does not consider several important issues such as dynamic composite services, mechanisms of admission control, distributed component resources or feedback to component providers to ensure the maintenance of service trustworthiness.

The combination of admission control, resource sharing and revenue maximisation are discussed by Urgaonkar [134]. He proposes an approach that combines admission control techniques with dynamic provisioning for shared resources to handle diverse workloads and maximise revenue. The admission control aims to remain operational under extreme overloads. It increases efficiency by classifying the requests into classes and admitting or rejecting sets of requests instead of individual requests. Other work on the admission control, resource sharing and revenue maximisation include that by Tsakalozos et al. [128] as well as Marbukh and Mills [129] described in Subsection 3.6.1.

In addition to the aspects discussed by Urgaonkar, García et al [133] add consumer differentiation in the service provision. They discuss requirements definition and analysis that the control mechanisms must fulfil in service exchange between enterprises including consumer differentiation and protection from overload. Some of the requirements they discuss include that the QoS control mechanisms must be able to manage multiple simultaneous services with different limitations and must be scalable. Their proposed design aims to support differentiation between categories of service consumers and protect against server overloads. The main aspect of QoS that the authors aim to optimise is the service response time. The work highlights a number of significant issues towards maintaining trustworthy services. However, it does not consider composite services and service execution workflows which are of vital importance in the services paradigm. The support for the provision of composite services and their control mechanisms must, in addition to the requirements specified by the authors, take into consideration important issues such as the specific workflow patterns of executions and the aggregation of the capacities of the CS components.

3.7 Summary

This chapter initially discussed the concept of trust and the progress made in relation to trust in distributed systems in general. The chapter then provided justifications for the thesis' research and discussed existing research contributions in the related topics. While there is a wide range of active research work that relates to the thesis we describe the main contributions in each of the thesis subtopics. Recent work on trust in services and compositions is often characterised by the lack or weak consideration of some important aspects. Table 3.1 contains a summary of main categories of contributions and their weaknesses in the related work. This thesis aims to address those weaknesses.

Table 3.1: Summary of Contributions and Weaknesses in the Related Work

Contribution	**References**	**General Weaknesses and Shortcomings**
Trustworthiness reporting and assessment of services	Malik and Bouguettaya [7, 61], Skopik [65, 66], Dragoni [67], Pranata et al. [68], Spanoudakis and LoPresti [69], Majithia et al. [70], Maximilien and Singh [71]	Composition of services and structure, complete subjectivity, attention to attribute heterogeneity, combination of design and runtime assessment, component importance
Trustworthiness in service selection and composition	Paradesi et al. [72], Hang and Singh [73], Li and Wang [74], Mehdi et al. [75], Malik and Bouguettaya [7, 61]	Evaluation of trustworthiness for components not available to consumers as standalone services, variations in service composition structure, attribute characteristics, hiding untrustworthy components
Distribution of trustworthiness updates from composite to component services	Nepal et al. [79], Wen et al. [81]	Collaboration between distributed compositions, attribute characteristics, composition structure, fairness
Attack surface based quantification of security	Howard et al. [101], Manadhata and Wing [102]	Composition of attack surfaces of multiple applications, range of security aspects and attributes
Alternative methods for security metrics and models	Jaquith [109], CVSS [98], Weinstock et al. [104], Microsoft STRIDE [106]	focus on threats and vulnerabilities, subjectivity, scale of required human intervention
Aggregation of QoS attributes for compositions	Cardoso et al. [110], Jaeger et al. [111], Hwang et al. [113]	Limited range of attributes, aggregation of reputation as sum or average from components
Reliability of component and composite services	Zheng and Lyu [82], Hwang et al. [113], Grassi and Patella [114], Moser et al. [30]	Other trustworthiness attributes, distribution of composition reliability to components
Maintaining trustworthiness and profitability	Tsakalozos et al. [128], Marbukh and Mills [129], Urgaonkar [134], García et al [133]	Range of trustworthiness attributes (e.g. not only response time), optimisation of pricing, composite services and structure

Chapter 4

Trustworthiness Monitoring and Prediction

4.1 Introduction

As described in the scenario in Section 2.5, CSPs need to select the most trustworthy component services available in the distributed service environment in order to carry out the composite service activities. The chapter presents an approach to the monitoring and prediction of trustworthiness of services that are assembled from component services. This chapter describes algorithms and models that are required to optimise the selection and to monitor the trustworthiness of a service composition.

The chapter discusses the aggregation of trustworthiness attributes for composite and component services into a common trustworthiness value. The techniques consider a number of criteria during the aggregation. One of such criteria is that component services in a composition may vary in their importance to the composite service as a whole. For example, in a travel service a user may not appreciate all component services to the same extent such as car rental, medical insurance, flight booking, etc. Therefore, it is more useful to see the composite service as a unit that is composed of unequal subunits in terms of their contribution to the trustworthiness of the service.

The components may differ also in the probability of their execution in their CS due to reasons such as limitations in their resource capacity.

The chapter is organised as follows. Section 4.2 explains techniques for the aggregation and calculation of the trustworthiness of a composite service based on the service's composition plan. It also includes calculation of service costs. Section 4.3 discusses the aggregation of trustworthiness attributes in order to collectively evaluate candidate services and constructs. The section also discusses the function and architecture of the trustworthiness software module and optimisation of service compositions. Section 4.4 describes experiments using simulations of the trustworthiness based service selection and composition trustworthiness computation. A summary is described in Section 4.5.

4.2 Trustworthiness Attributes and Aggregation

In this work, the trustworthiness value T_{cs} of a composite service is modelled in general as a function g of the trustworthiness of its components:

$$T_{cs} = g\left(\{T_1, T_2, ..., T_m\}\right) \tag{4.1}$$

However, the calculation of the trustworthiness value depends on the trustworthiness attributes and the structure of the business process. The selection of component services statically (during design time) or dynamically is based on the predicted trustworthiness value of the composite service. Trustworthiness attributes include a set of attributes that are used to determine the overall trustworthiness value as discussed in Subsection 4.2.1.

Selected services are executed in a business process. The process is viewed externally as a Web service. The computation of the trustworthiness of the CS depends on the way the abstract service is constructed. Common business process constructs are discussed in Section 2.4. The computation also depends on the probability of execution

and the importance of the component services in the composition. Component services in a composition may vary in their importance to the composite service as a whole.

The prediction of the CS trustworthiness depends on the way the process is constructed. It also depends on the probability of execution and the importance of the components in the composition. For example, in a travel service a user may not appreciate all component services to the same extent such as car rental, medical insurance, flight booking, etc. The probability of execution of a component service may be based on the characteristics of the process or on limited supply of the component service. For example, in an emergency CS a fire or ambulance service may be required in an estimated percentage of executions. An example of limited capacity is where a certain car rental service is most trustworthy but has limited supply. In that case more demand requires additional supply from other possibly less trustworthy car rental service providers.

4.2.1 Trustworthiness Attributes

The trustworthiness value of a CS depends on its trustworthiness attributes and the structure of the business process. In Chapter 3, we discussed attributes investigated in literature that are identified under the categories of QoS, security and reputation. However, the discussed attributes cannot be exhaustive. The identification of a set of attributes suitable for a particular business activity or for a service environment depends on context, requirements and possibly other factors. In this chapter, we consider a set of common trustworthiness attributes that can affect the overall trustworthiness of a CS.

- *Reliability* (r): the rate of successful executions of a service without full or partial failures per total number of executions ($0 \leq r \leq 1$).
- *Uptime* (a): the percentage of time of availability of a service for the admission of requests over the total measurement time ($0 \leq a \leq 1$). Uptime is used as a synonym for availability.
- *Reputation* (p): the data available about a service from consumer satisfaction ratings. In this thesis we consider the reputation as a value p where $0 \leq p \leq 1$.

- *Security* ($\mathcal{S}$): includes a number of attributes such as encryption, confidentiality, non-repudiation and authentication. A security attribute σ_i for a service s_i is a boolean $\sigma_i \in \{0, 1\}$ with 1 representing the fulfilment of the attribute and 0 for non-fulfilment. For example in encryption the fulfilment means the messages are securely encrypted with at least a minimum allowed key length.

- *Response time* (t): the response time of a service is used as a metric of performance. After aggregation of the component response times, the CS response time is measured against required response time in the service contract.

- *Capacity (y)*: is the number of executions that can be performed simultaneously. The aggregation of the capacity may result in an overall CS capacity that does not fulfil the requirements of the service contract.

- *Cost* (c): monetary cost of a service. Cost is sometimes considered as a trustworthiness attribute [e.g. 135]. However, we include in the aggregation techniques in any case as it is an important factor in service composition optimisation.

4.2.2 Importance Weight of Components

Each component s_i in a composition has a weight ω_i based on its importance to the reputation of the composition $\omega_i \in \{\omega_1, ..., \omega_l\}$ where $0 \leq \omega_i \leq l$, l is the number of component services excluding alternatives in exclusive choice constructs and $\sum_{i=1}^{l} \omega_i = l$. We consider components in an Exclusive Choice construct as a single unit in terms of their weight ω_θ and its calculation, where θ is the service construct. For example, consider the case where a requirement may be satisfied by only one of two services $\{s_1, s_2\}$ and the trustworthiness of s_1 is more than that of s_2 but its capacity is limited to a certain quantity. When s_1 becomes fully in use, s_2 is invoked. Therefore, a common weighting value is used. For a composition with m components and x exclusive choice constructs:

$$l = m - (n_x - x) \tag{4.2}$$

where n_x is the total number of components in the x constructs.

The weighting of the components is used in the calculation of the CS reputation, such as in the case of a Sequence with n components:

$$p_\theta = \prod_{i=1}^{n} {p_i}^{\omega_i} \tag{4.3}$$

In order to differentiate between mandatory and optional components, a weight threshold Ω is set. A component service with $\omega_i < \Omega$ is considered optional and can be excluded from the CS execution when necessary for instance due to the unavailability of the component. Optional components can be excluded also from the aggregation of other trustworthiness attributes such as capacity. The component weights are useful when the capacity of some components are in full usage or close to becoming so. In that case non-critical components can be excluded from the CS execution. This is particularly useful if the request would otherwise be rejected or when low remaining component resources can be saved for prioritised requests. More details on this is in Subsection 7.3.2.

4.2.3 Aggregation of Attributes

Table 4.1 shows our functions for calculating the considered trustworthiness attributes per service construct. The following discussion details the approaches for their aggregation. We use θ to denote a service construct in a composition.

For the purpose of trustworthiness attribute aggregation, a CS is represented as a hierarchy of constructs. In order to aggregate the values of an attribute, the workflow undergoes a series of reductions until the highest level construct is reached. The innermost constructs are reduced first at each step. The type of the final construct depends on the outermost pair of component services or gateways (i.e. start and end services or gateways). Figure 4.1 illustrates the reduction of the workflow of a composite service containing a variety of construct types. The CS is reduced to a final sequence construct in step (3). A reduced construct is treated like a component service during attribute aggregation.

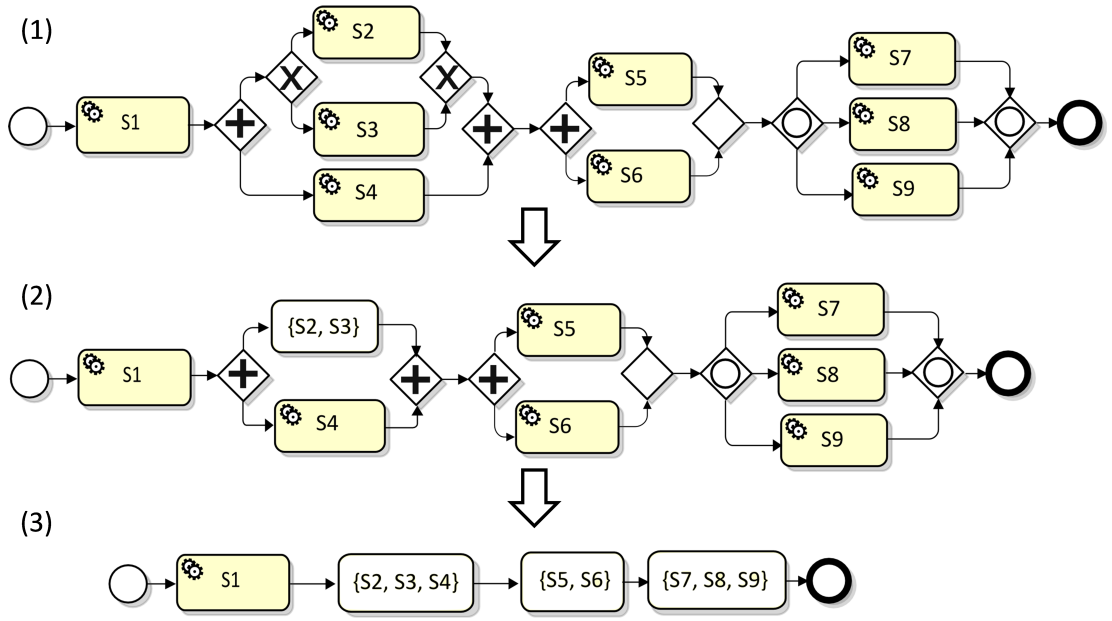


Figure 4.1: Hierarchical Reduction of CS Constructs

4.2.3.1 Reliability

(i) *Sequence*, *Synchronized Parallel* and *Unordered Sequence*: A failure of a component means failure of subsequent dependent components. This is unlike some other types of constructs (e.g. Unsynchronized Parallel) where subsequent components may be partially independent of the failure of the construct components and can be executed as long as a minimum set of components succeeds. Therefore, we calculate the reliability of the CS as a product of that of its components.

$$r_\theta = \prod_{i=1}^{n} r_i \tag{4.4}$$

(ii) *Loop*: The reliability of a Loop containing n iterations of a service s_i is the same as a Sequence construct of n copies of s_i i.e. $r_\theta = {r_i}^n$.

(iii) *Exclusive Choice*: The reliability this construct is the sum of that of the exclusive components multiplied by their probabilities of execution in the CS.

(iv) *Unsynchronized Parallel*: Since an Unsynchronized Parallel construct only fails if all constituent services fail, its reliability is calculated as follows.

$$r_\theta = 1 - \prod_{i=1}^{n}(1 - r_i) \tag{4.5}$$

(v) *Multi-choice with Synchronized Merge*: In each subset of components that may be executed in parallel, all its components must be executed successfully. Therefore, we sum the probabilities of each subset multiplied by the reliability of that subset. In a construct θ with a set S of components and two or more probable subsets of components that may be executed in parallel, its reliability is calculated as follows.

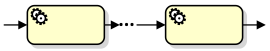
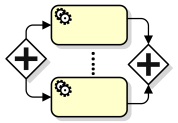
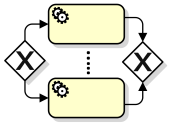
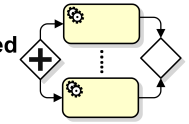
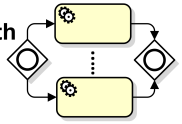
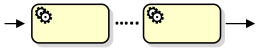
$$r_\theta = \sum_{j \subset S} (\rho_j \cdot \prod_{i \in j} r_i) \tag{4.6}$$

4.2.3.2 Uptime

Reliability and uptime aggregations are similar. However, the equations in case of uptime apply only to mandatory components. The downtime of any mandatory component in these constructs results in the unavailability of the whole construct because of the dependency between their components. As described in Subsection 4.2.2 a mandatory component is one whose importance weight is above threshold. An optional component is excluded from execution when unavailable. Its uptime does not affect the uptime of the construct. Therefore, Eq. (4.7) for Sequence and Synchronized Parallel is subject to $\omega_i < \Omega$.

$$a_\theta = \prod_{i=1}^{n} a_i \tag{4.7}$$

Table 4.1: Aggregation of Trustworthiness Attributes and Cost per Process Construct

Construct	Reliability (r_θ)	Reputation (p_θ)	Encryption ($\mathfrak{d}_\theta$)	Resp. Time (t_θ)	Capacity (y_θ)	Cost (c_θ)
Sequence 	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} p_i^{\omega_i}$	$\min_{i=1}^{n} \mathfrak{d}_i$	$\sum_{i=1}^{n} t_i$	$\min_{i=1}^{n} y_i$	$\sum_{i=1}^{n} c_i$
Synchronized Parallel 	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} p_i^{\omega_i}$	$\min_{i=1}^{n} \mathfrak{d}_i$	$\max_{i=1}^{n} t_i$	$\min_{i=1}^{n} y_i$	$\sum_{i=1}^{n} c_i$
Loop 	$r_i^{\,n}$	$p_i^{n \cdot \omega_i}$	$\mathfrak{d}_i$	$n \cdot t_i$	y_i	$n \cdot c_i$
Exclusive Choice 	$\sum_{i=1}^{n} \rho_i \cdot r_i$	$(\sum_{i=1}^{n} \rho_i \cdot p_i)^{\omega_\theta}$	$\min_{i=1}^{n} \mathfrak{d}_i$	$\sum_{i=1}^{n} \rho_i \cdot t_i$	$\sum_{i=1}^{n} \rho_i \cdot y_i$	$\sum_{i=1}^{n} \rho_i \cdot c_i$
Unsynchronized Parallel 	$1 - \prod_{i=1}^{n}(1 - r_i)$	$\prod_{i=1}^{n} p_i^{\omega_i}$	$\min_{i=1}^{n} \mathfrak{d}_i$	$\min_{i=1}^{n} t_i$	$\max_{i=1}^{n} y_i$	$\sum_{i=1}^{n} c_i$
Multi-choice with Synchronized Merge 	$\sum_{j \subset S} (\rho_j \cdot \prod_{i \in j} r_i)$	$\sum_{j \subset S} (\rho_j \cdot \prod_{i \in j} p_i^{\omega_i})$	$\min_{i=1}^{n} \mathfrak{d}_i$	$\sum_{j \subset S} (\rho_j \cdot \max_{i \in j} t_i)$	$\sum_{j \subset S} (\rho_j \cdot \min_{i \in j} y_i)$	$\sum_{j \subset S} (\rho_j \cdot \sum_{i \in j} c_i)$
Unordered Sequence 	$\prod_{i=1}^{n} r_i$	$\prod_{i=1}^{n} p_i^{\omega_i}$	$\min_{i=1}^{n} \mathfrak{d}_i$	$\sum_{i=1}^{n} t_i$	$\min_{i=1}^{n} y_i$	$\sum_{i=1}^{n} c_i$

θ = construct, n= no. of construct components, ρ_i= probability of execution of component s_i,

ρ_j= probability of execution of subset j of construct components

4.2.3.3 Reputation

The aggregation of reputation of components follows a similar approach to that of reliability but with consideration of the weights of components in the construct with some differences in the aggregation formulas for some constructs. This approach is based on our assumption that the reputations of the components of a CS are interdependent and that the reputation of a CS is influenced by the importance of each component as well as its reputation.

(i) *Sequence*, *Synchronized Parallel* and *Unordered Sequence*: The reputation is calculated as a product of that of constituent services taking the service importance into consideration as in Equation (4.3).

(ii) *Loop*: The reputation of a Loop containing n iterations of a service s_i is the same as a Sequence of n copies of s_i i.e. ${p_i}^{n \cdot \omega_i}$.

(iii) *Exclusive Choice*: Each service s_i among the alternative services in an Exclusive Choice has a probability ρ that it will be executed and $\sum_{i=1}^{n} \rho_i = 1$. As described earlier an Exclusive Choice is considered one unit in the CS component reputation weights. Therefore, we aggregate reputation of the construct as sum of component reputations multiplied by their probabilities of executions.

$$p_\theta = (\sum_{i=1}^{n} \rho_i \cdot p_i)^{\omega_\theta} \tag{4.8}$$

(iv) *Unsynchronized Parallel*: Since all component services are executed, the reputation takes all services into consideration as in Equation (4.3).

(v) *Multi-choice with Synchronized Merge*: In this construct, the execution of each subset j of all possible subsets of the set S of construct services ($j \subset S$) is associated with a probability ρ_j that it will be executed where

$$\sum_{j \subset S} \rho_j = 1 \tag{4.9}$$

The construct reputation considers both the probability of execution and weighting of component services:

$$p_\theta = \sum_{j \subset S} (\rho_j \cdot \prod_{i \in j} p_i^{\omega_i}) \tag{4.10}$$

4.2.3.4 Security

For security attributes, the level of security for an attribute in a composition follows the weakest link principle for all the CS components (see e.g. [136] on the weakest link principle). Table 4.1 illustrates the aggregation for encryption $\mathfrak{d}_\mathrm{i}$ as an example security attribute. Suppose an attribute $\sigma_{k,i}$ is one of z evaluated (i.e. verified compliant/non-compliant) security attributes of components of a CS; for a component s_i the attribute $\sigma_{k,i} \in \{0, 1\}$ and $\sigma_{k,i} \in \{\sigma_{1,i}, \sigma_{2,i}, ..., \sigma_{z,i}\}$. We calculate the value for an attribute $\sigma_{k,cs}$ for a CS with m components $\sigma_{k,cs}$ as:

$$\sigma_{k,cs} = \min_{i=1}^{m} \sigma_{k,i} \tag{4.11}$$

To aggregate the values of all security attributes in a composition and calculate the overall level of security ($0 \leq \mathcal{S}_{cs} \leq 1$) based on z attributes we first take the weighted sum $\hat{\sigma}_{cs}$ of the verified attributes:

$$\hat{\sigma}_{cs} = \sum_{k=1}^{z} \gamma_k \cdot \sigma_{k,cs} \tag{4.12}$$

where γ_k ($\gamma_k \geq 0$) is the weight for the attribute $\sigma_{k,cs}$. It sets the effect of the attribute on the security level and trustworthiness. The value of γ_k depends on multiple factors including the number of security attributes considered, the priority of each attribute, potential resulting attack graphs in the CS and the likelihood of the related vulnerabilities being exploited. We propose the following formula for the level of security:

$$\mathcal{S}_{cs} = 1 - e^{-\hat{\sigma}_{cs}} \tag{4.13}$$

The value of γ_k should meet the following requirements:

- when all the CS security attributes are fulfilled (i.e. $\sum_{k=1}^{z} \sigma_{k,cs} = z$) Eq. (4.13) should result in a security level $\mathcal{S}_{cs} \approx 1$, and
- when a attribute that cannot be compromised is not fulfilled in the CS, $\mathcal{S}_{cs}$ should fall below a preset security threshold (e.g. 0.95).

An alternative more comprehensive approach to security evaluation using attack surface based security metrics is discussed in Chapter 5.

4.2.3.5 Response Time

(i) *Sequence* and *Unordered Sequence*: In both of these constructs the response time is summed to provide the total time of the construct execution; $t_\theta = \sum_{i=1}^{n} t_i$.

(ii) *Synchronized Parallel*: The components are executed in parallel but the next construct cannot commence its execution until all parallel components are complete. Therefore, construct response time equals that of the longest of its components' response times i.e.

$$t_\theta = \max_{i=1}^{n} t_i \tag{4.14}$$

(iii) *Loop*: The response time of a loop is the number of executions by its component's response time.

(iv) *Exclusive Choice*: We use the execution probability-based weighted average of the components response times as the construct's average response time.

(v) *Unsynchronized Parallel*: The next construct starts execution once the first executing component in this construct completes. Therefore, the minimum of the components response times is the construct response time.

(vi) *Multi-choice with Synchronized Merge*: The response time for each subset j with execution probability ρ_j equals the longest of its components response times.

Subsequently, as in the Exclusive Choice we take the weighted average of the subsets' response times.

$$t_\theta = \sum_{j \subset S} (\rho_j \cdot \max_{i \in j} t_i) \tag{4.15}$$

4.2.3.6 Capacity

(i) *Sequence*, *Synchronized Parallel* and *Unordered Sequence*: The capacity of each of these constructs equals the minimum capacity among its components when buffering is not taken into consideration.

$$y_\theta = \min_{i=1}^{n} y_i \tag{4.16}$$

(ii) *Loop*: Since the same component is executed sequentially, the construct's capacity is the same of that of the component i.e. $y_\theta = y_i$.

(iii) *Exclusive Choice*: The construct capacity equals the execution probability-based weighted average of the capacities of exclusive components.

(iv) *Unsynchronized Parallel*: At least one component of this construct is required to be executed. Therefore, its capacity equals the maximum component's capacity.

$$y_\theta = \max_{i=1}^{n} y_i \tag{4.17}$$

(v) *Multi-choice with Synchronized Merge*: All components in each subset j of the construct components with probability of execution ρ_j must have the capacity to execute the CS. Therefore, the minimum of their capacity provides the capacity of the subset. The capacity of the construct is the total of product of the subset capacities by their probabilities.

The capacity aggregation methods above do not describe how to consider buffering. Concisely, we set a threshold for the CS execution queue size. The threshold value is based on constructs capacities, their response times and the total allowed CS response

time. The buffering maximises the usage of the components capacities and significantly increases the capacity of the CS.

4.2.3.7 Cost

(i) *Sequence*, *Synchronized Parallel*, *Unsynchronized Parallel* and *Unordered Sequence*: Since all components in the constructs are executed, their cost is the sum of the cost of all components.

(ii) *Loop*: The cost of a loop construct of n iterations of a service s_i is the same as a Sequence construct of n copies of s_i i.e. $n \cdot c_i$.

(iii) *Exclusive Choice*: Since each service among the alternative services in the construct has a probability ρ that it will be executed and $\sum_{i=1}^{n} \rho_i = 1$, its cost is the probability-based weighted average of that of each component service.

(iv) *Multi-choice with Synchronized Merge*: The calculation of cost in this construct is the execution probability-based weighted average of the cost of execution of each subset of components.

$$c_\theta = \sum_{j \subset S} (\rho_j \cdot \sum_{i \in j} c_i) \tag{4.18}$$

4.2.4 Trustworthiness Update Procedure

An algorithm is proposed here to predict and update the trustworthiness value of service. The algorithm is faster than those proposed for multiagent systems in REGRET [37] and FIRE [38] since there is no need to recursively run through all the ratings with each new rating received. We discuss reputation as a trustworthiness attribute in the algorithm. However, the same procedure can be applied to other attributes. In this algorithm, the reputation is determined using moving averages that are updated with every new rating. Older ratings reduce in value over time. The comparison with other algorithms is further discussed in the evaluation.

The reputation of a service s_i is determined by two values; the reputation value p_i, $0 \leq p_i \leq 1$ and the *confidence* f_i in the value and $0 \leq f_i \leq 1$. Both of the two values (i.e. reputation and confidence values) are important in indicating the status of a composite and component services. Reduction of the reputation value signifies receiving consistent bad ratings of the service while reduction in confidence indicates either low number of ratings received recently, significant fluctuations in the rating values or both. Those fluctuations may for example indicate that a service is not scalable enough to meet demands during peak times. The reputation value p_i is calculated as a dynamically weighted moving average of the service's rating values. When a new rating is received the reputation value is updated.

4.2.4.1 Weights of Ratings

First, a value representing the *accumulated weight* w_i of all received ratings for a service is updated. This weight is based on the *recency* and the *category* of the ratings. Recency weight w_t indicates how recent are the ratings received for the service. The more recent the ratings the higher the weight because future ratings are more likely to be close to the latest ratings. Reputation ratings of a service can be classified into a set of categories or types with different weights depending on the way they are gathered. Examples of types of consumer ratings may include feedback on satisfaction, support, speed, etc. A service's customers might not appreciate those categories equally and hence the customisable category weighting w_g.

Recency weight w_t decays exponentially and $0 \leq w_t \leq 1$, as follows.

$$w_t = e^{-\lambda \cdot \Delta t} \tag{4.19}$$

where λ is the decay constant; a customisable positive number that controls the rate of decay, and Δt in relation to a single rating is the age of that rating i.e. the difference between the current time and the time when the rating took place, while Δt for the reputation value from the latest update is the age of the last update of the accumulated weight w_i.

The accumulated weight of the reputation value w_i ($w_i > 0$) is updated when a new rating value P is received, as follows.

$$w_i \leftarrow w_i \cdot w_t + w_P \tag{4.20}$$

where w_P is the weight of the new rating value P which is calculated as follows.

$$w_P = w_{t_P} \cdot w_{g_P} \tag{4.21}$$

where w_{t_P} and w_{g_P} are the recency weight and category weight for the rating P respectively. The value w_{t_P} is calculated as in Equation (4.19).

For a rating that is generated at the time of calculation (i.e. $\Delta t = 0$ and $w_P = w_{g_P}$), the new accumulated weight:

$$w_i \leftarrow w_i \cdot w_t + w_{g_P} \tag{4.22}$$

4.2.4.2 Update of Reputation

The following is the formula for updating p_i after receiving a new rating.

$$p_i \leftarrow \frac{(w_i - w_P) \cdot p_i + w_P \cdot P}{w_i} \tag{4.23}$$

To facilitate the recalculation of the trustworthiness value when new ratings are received, the values of w_i and p_i are stored after each update.

4.2.4.3 Confidence

Since confidence reflects both the frequency of receiving new ratings and the stability of their values as described earlier, the confidence value of service s_i, f_i is calculated as:

$$f_i = f_\eta \cdot f_\delta \tag{4.24}$$

where f_η is called the *ratings quantity confidence* indicating how frequent new ratings are received; and f_δ the *ratings quality confidence* which indicates the stability of the ratings' values. The more frequent and stable the ratings the more the confidence i.e. certainty in relation to the calculated reputation value. The following formula calculates f_η:

$$f_\eta = 1 - e^{-\alpha \cdot w_i} \tag{4.25}$$

where α is a constant parameter that can be used to adjust the slope of the relationship between the sum of the ratings' weights and the quantity confidence. The higher the value of α the faster the full confidence (i.e. 1) is reached. It can be set to any positive value but for gradual increase in confidence it should typically be set to a value between 0 and 1. The confidence increases in proportion to the number of ratings and to the degree of their recency.

The quality confidence f_δ is calculated as follows.

$$f_\delta = 1 - d_i \tag{4.26}$$

where d_i is the deviation history of the ratings around the reputation value, calculated as in Equation (4.27).

$$d_i \leftarrow \frac{(w_i - w_P) \cdot d_i + w_P \cdot |p_i - P|}{w_i} \tag{4.27}$$

To help update the reputation when new ratings are received, the value of d_i is stored after each update. The result from $|p_i - P|$ is the absolute difference between the overall reputation value and individual rating value. The value of f_δ indicates the deviation of the ratings around the overall reputation value and ranges between 0 (highest deviations) and 1 (lowest deviations).

4.2.4.4 Data Source and Rater Credibility

The credibility of raters is used to protect from malicious and biased raters and data sources. The main existing approaches in dealing with the problem are through majority

rule and rating of consumers and data sources [7, 60] as discussed in the literature review Subsection 3.2.3. Reputation ratings that are sent to the trustworthiness module (see Subsection 4.3.2) are required to include ConsumerID and TransactionID fields in order to protect from vulnerabilities such as slandering (see Subsection 3.2.3 on threats to trust and reputation systems).

We propose the use of majority rule approach for the computation of rater credibility. When such variable is included in the above algorithm it affects the weights of new received ratings. In that case Eq. (4.21) includes the rater credibility β as in Eq. (4.28).

$$w_P = w_{t_P} \cdot w_{g_P} \cdot \beta \tag{4.28}$$

where $0 \leq \beta \leq 1$. It will consequently affect the level of influence of the new rating on the updated reputation and confidence.

4.3 Service Trustworthiness and Selection

4.3.1 Aggregated Trustworthiness

A trustworthy composite service may incorporate a set of components that are collectively trustworthy but their attribute values vary. Therefore, considering trustworthiness attributes individually and setting their threshold can exclude some more trustworthy components or constructs than those selected. A unified view of trustworthiness during selection can have advantages as it weighs all attributes together in the selection decision. However, multidimensional trustworthiness is a complex concept as it involves heterogeneous attributes where, for example, some can be affected by the dependency between components e.g. reliability, while others are not interdependent e.g. response time. Additionally, attributes are affected in different fashion by the types of paths in CS processes. Therefore, any combination of those attributes into a common value or ranking system can only be an approximation.

Multi-Criteria Decision Making approaches have been suggested to rank services or constructs according to their attributes. Malik and Bouguettaya [7] suggest the use of *Simple Additive Weighting* citing that it provides results comparable to more sophisticated methods. Zia ur Rehman et al. [122] describe a method called *Exponential Weighted Difference* that restricts the effect of mutual cancellation between criteria (attribute values) exceeding and below requirements.

Each trustworthiness attribute has a computed or monitored value and a required value. The required value is based on consumer expectation and/or CS provider requirements. In order to perform common mathematical operations on the attributes and rank services based on their collective trustworthiness, the attributes need to be normalised. Additionally, prioritisation of the attributes ensures that the more important attributes to service trustworthiness are given more weight in the decision.

Normalisation of attributes to obtain attribute values using the same measurement units, requires that we determine the maximum and minimum allowed values for each attribute. Attributes differ in their measurement units e.g. continuous vs. discrete value, and in their optimal values i.e. maximum vs. minimum. For example, for reliability the minimum value is 0 and the maximum is 1 (optimal). Defining maximum and minimum values may not be as straightforward for some attributes. For example, the minimum response time (optimal) can be set to approximately 0 while the maximum can be defined as the maximum acceptable response time before considering the execution as unsuccessful. Additionally, some attributes may be discrete or binary such as compliance attributes particularly those that are security related such as non-repudiation and integrity.

Equation (4.29) shows how to determine the normalised value $\tau_{h,norm}$ for an attribute τ_h of a service s_i with N attributes depending on their optimal value, $\tau_h \in \{\tau_1, ..., \tau_N\}$.

$$\tau_{h,norm} = \begin{cases} \frac{\tau_{h,max} - \tau_{h,val}}{\tau_{h,max} - \tau_{h,min}} & if\ \tau_{h,min}\ is\ optimal \\ \\ \frac{\tau_{h,val} - \tau_{h,min}}{\tau_{h,max} - \tau_{h,min}} & if\ \tau_{h,max}\ is\ optimal \end{cases} \tag{4.29}$$

where $\tau_{h,val}$ is the actual attribute value, $\tau_{h,max}$ is the maximum, $\tau_{h,min}$ is the minimum value and $\tau_{h,max} \neq \tau_{h,min}$. The equation is applicable also when the type of the attribute values are binary or discrete.

Selection candidate services or constructs can then be ranked based on their values. A value τ_θ for a construct θ is calculated using the weighted average of the difference between normalised attributes and their specified thresholds $\tau_{i,req}$.

$$\tau_\theta = \frac{1}{N}\sum_{h=1}^{N} \lambda_h(\tau_{h,norm} - \tau_{h,req}) \tag{4.30}$$

where $(\tau_{h,norm} - \tau_{h,req}) \geq 0$, λ_h is the weight of attribute τ_h, $\lambda_h \geq 0$ and $\sum_{h=1}^{N} \lambda_h = N$. The value τ_θ can be easily standardized to a value between 0 and 1 to represent the trustworthiness value. In case of attribute values above thresholds, the difference $(\tau_{h,norm} - \tau_{h,req})$ can be reset to 0 to prevent mutual cancellation.

4.3.2 Trustworthiness Module

A trustworthiness software module is developed for the runtime monitoring and prediction of composite service trustworthiness based on a set of mechanisms and metrics to ensure contract compliance. Monitoring is the process of checking that service contracts are fulfilled over time, particularly if changes can occur to operational or business environments or to internal service quality, security or reputation. Monitoring is also used to detect vulnerabilities and discover attacks on a service, e.g. by making use of intrusion detection systems or dynamic testing tools available in the environment.

A composite service provider is a service provider that is responsible for constructing service compositions and offering them to consumers. A CSP is notified of important changes in the trustworthiness of the composite service as a result of one of its components. A component service that is below the satisfactory trustworthiness value can be replaced with another component service offering the same functionality but with better trustworthiness. The monetary cost of a composite service as a result of

its adaptation is also determined. The consideration of costs ensures that a balance is maintained between both trustworthiness and cost efficiency of the service.

Figure 4.2 shows the architecture of the trustworthiness module. The trust events refer to the notifications received by the module from external components such as event processing, QoS monitoring, consumer ratings, security testing tools and other components. Those events include metrics and alerts that indicate violations or adherence to the service contracts, threats or changes in the environment. In addition to the direct experience through those events, the trustworthiness module can exchange recommendations with other online modules in relation to service trustworthiness. Composition plans and existing composite services can be evaluated by the module and their trustworthiness values are calculated based on received BPMN models of the planned or existing CS in XML format.

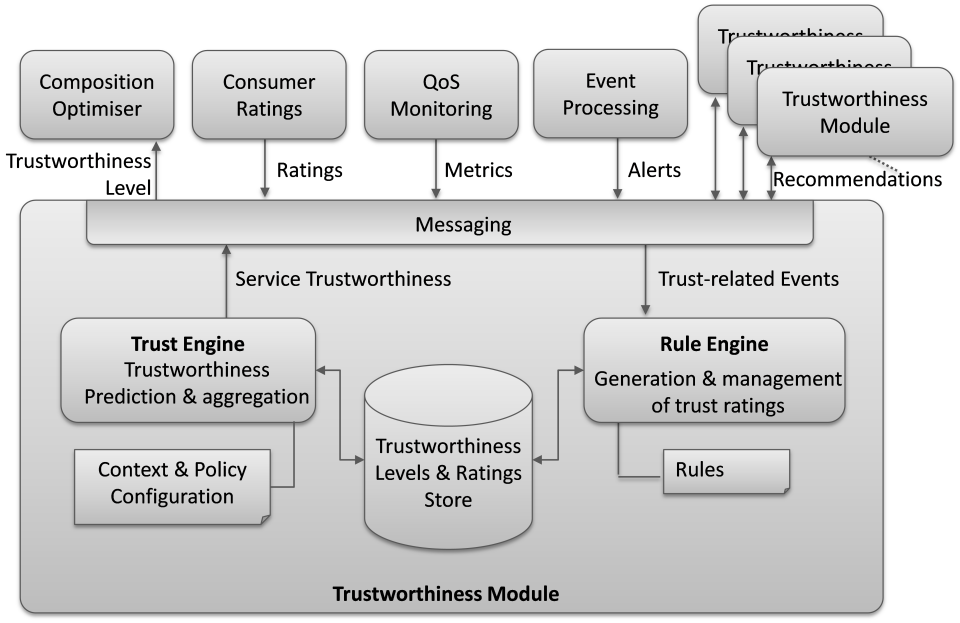


Figure 4.2: Trustworthiness Module

Incoming events are evaluated by a rules engine to generate trustworthiness attribute ratings. The rules calculate the rating for the event and add other properties including the event timestamp and the concerned trustworthiness attribute. Trustworthiness ratings are then stored by the module and can be used for calculating and updating

the overall trustworthiness value of each service and its composite services. The trustworthiness value can be used by other components to optimise or evaluate the trustworthiness of composite services. Context configurations allow customisation of the trust context by adjusting the weighting of trustworthiness attributes e.g. security and performance attributes. Policy configurations allow setting the trustworthiness thresholds and algorithmic constants such as the rating decay rate. The trust engine is responsible for the aggregation of trustworthiness of a composite service from that of its components and providing a prediction of the trustworthiness value of a service.

The trustworthiness module is implemented in Java as dynamic OSGi service platform [137] sub-modules and uses Drools [138] for implementing the rating rules. This architecture allows the substitution of the sub-modules dynamically as in the case where alternative algorithms are required or configurations for the policy and context need to be changed.

4.3.3 Optimal Service Composition

For optimal selection of a component service for service compositions, the following formula is used:

$$\max\left(\omega_T \cdot T_{cs} + \frac{\omega_c}{C_{cs}}\right) \tag{4.31}$$

where C_{cs} is the cost of the CS and T_{cs} is a representation of the trustworthiness calculated from security, reliability and reputation value based on Equation (4.30). Values ω_T and ω_c are constants used to normalise the values of trustworthiness and cost respectively and to customise their priority.

In order to optimise service selection allowing to choose among the best component services as per the computation techniques described in this chapter, an optimisation solution is needed. Since the trustworthiness values and costs of component services have discrete values and because of the non-linearity of those attributes, linear programming and other solutions that require continuous variables and/or linearity are not suitable. Additionally, the number of services to select from may be large making

heuristic methods a better option to provide fast and adequate results. Genetic algorithms are well-suited to these kinds of problems. A custom GA is required to suit the characteristics of the problem of service composition.

4.4 Simulation and Experiments

4.4.1 Description of GA

A custom GA is developed in MATLAB in which the fitness function uses Equation (4.31) together with the aggregation techniques that depend on the structure of the service composition. As illustrated in Figure 4.3 the genome is represented by a binary matrix where each row represents an ordered set of (concrete) services belonging to a single service type (abstract service). In each round, a selected service is represented by 1 and an unselected by 0. Therefore, each row must have only single 1 as only one service can be selected from each type to become a component of the composite service. Each type (row in the representation) is associated with a construct in the composite service and with a weighting value. Since the number of available services may be different for each service type, the number of columns in the matrix equals the size of the largest set of services belonging to one service type S where there are m service types.

$$\max_{i=1}^{m}(size(S_i)) \tag{4.32}$$

Empty elements in smaller sets are filled with Not-a-Number (NaN).

A set of matrices (using MATLAB cell array) are created as an initial population. The custom crossover function takes the parents as cell arrays, and returns the children that result from a two-point crossover by exchanging randomly selected sections in the parents' matrices. The custom mutation function randomly selects two elements in a row of a parent and swaps their values. Since all elements except one are set to 0, the mutation may have an effect only if the value of one of the affected elements equals 1. The number of generations can be fixed to a constant number or set to be

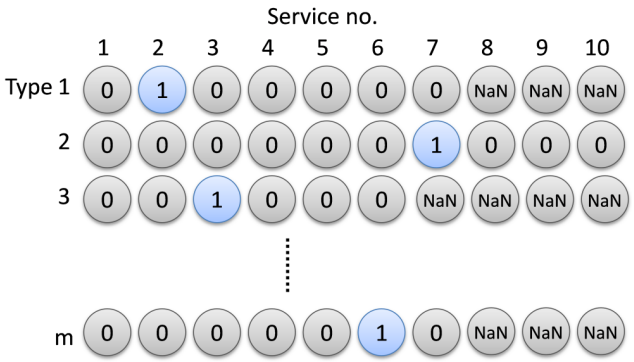


Figure 4.3: GA Genome

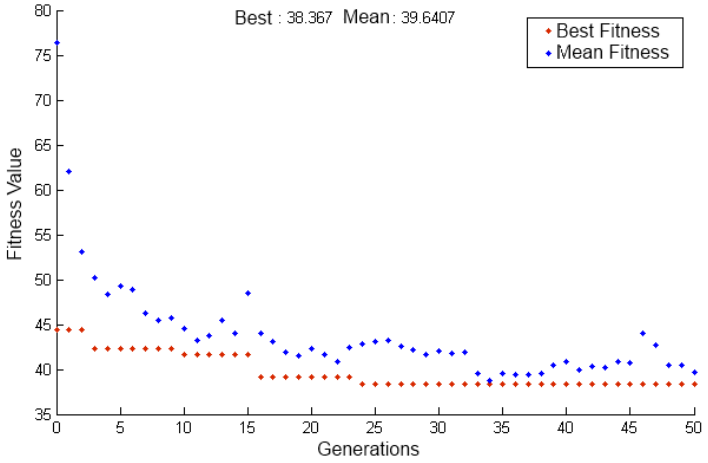


Figure 4.4: Scores by Fitness Function

proportionate to the number of service types and number of services. Figure 4.4 shows the improvement of the score of best composition over 50 generations for a simulation of services. Note that the problem is converted to a minimisation one. In the simulation there are 10 types of services organised in constructs as illustrated in Figure 4.1 with each type having between 5 and 10 concrete services. The cost and trustworthiness of the services are randomly assigned.

4.4.2 Comparison to Other Approaches

Figure 4.1 illustrates an example composite service used in the simulation that includes the composition constructs discussed earlier. Simulation services continue to receive new ratings over the duration of their runtime. The arrival time for service requests is based on Poisson distribution and the mean for the requests changes over time peaking

towards the end. Ratings are created based on results of service executions and their values vary between services, the time of the rating and whether there is an increased demand. The high demand is set to cause consistent low performance (resulting in mainly low reputation value) or fluctuations in performance (resulting mainly in low reputation confidence) in some of the services. A Gaussian random number generator is used to generate new ratings where the mean and the variance depend on the component service, its type (functionality), and the time of rating (e.g. high demand).

During the simulations each of the services receives a rating after each request. The ratings trigger the update and checking of the trustworthiness of the composite service. Figure 4.5 shows the evaluation of the trustworthiness of the services using our approach compared to that using other approaches including:

- averaging of the reputation of components as proposed by Hwang et al. [112].
- taking the minimum reputation of the components as the reputation of the composite service based on the weakest link principle where the reputation of the composition is as good as that of its component with the lowest reputation.

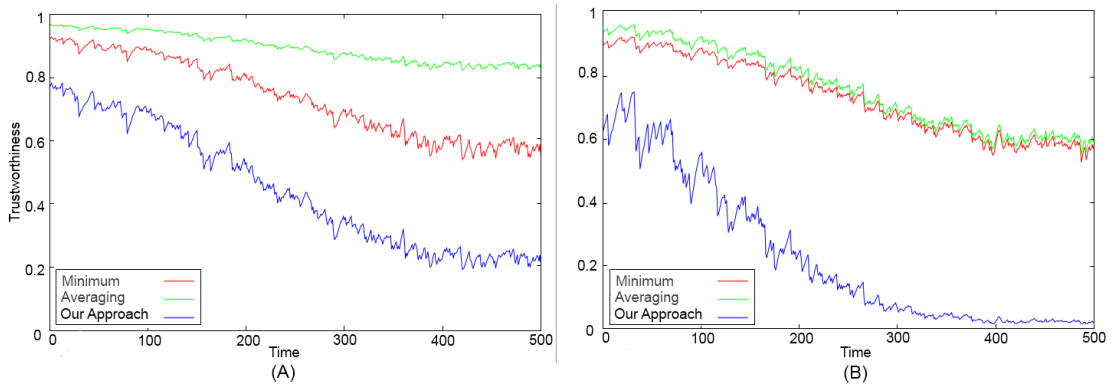


Figure 4.5: Comparison between Approaches to Trustworthiness Aggregation

In Figure 4.5 (A) only three out of ten component services significantly decrease in their trustworthiness during the peak time. Despite the low reputation of three component services, the reputation calculated using the averaging technique is still high. The weakest link technique only shows the lowest trustworthiness component but does not reflect information on other components with weak reputation while our approach

maintains better view of the state of the CS. In Figure 4.5 (B) all component services' trustworthiness values decline significantly. The weakest link technique estimates trustworthiness of the CS around the same value as in previous case despite the decline of trustworthiness of all component services. The averaging technique also does not reflect the low reputation of every component service. The trustworthiness based on our approach falls to a very low level indicating the low trustworthiness of the composition.

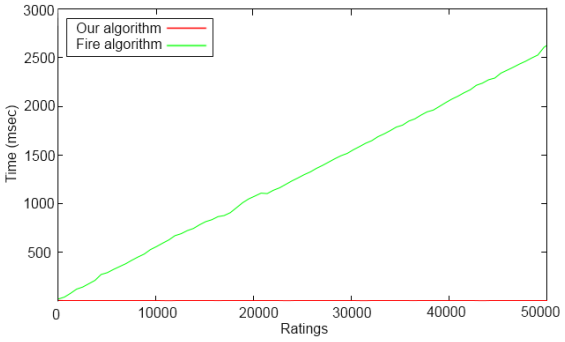


Figure 4.6: Processing Time in msec for FIRE and Our Algorithm

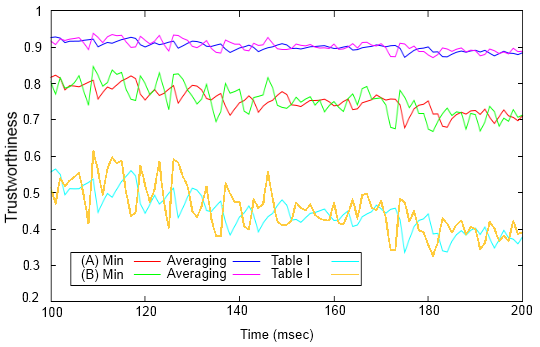


Figure 4.7: Effect of Processing Time on Trustworthiness Calculations

FIRE [38] is a widely cited trust management model and algorithm for the assessment of the trustworthiness of agents in open multiagent systems. FIRE extends REGRET system developed by Sabater [37]. Unlike our approach of using a moving average, FIRE algorithm recursively runs through all the ratings whenever a new rating is received. This results in an increasing delay in responding to requests for trustworthiness evaluation as ratings increase in quantity.

Figure 4.6 compares the processing times of new ratings required by the algorithm described in this chapter and that of FIRE. The figure clearly shows our algorithm is considerably faster as the number of ratings available for assessment increases. Note that re-assessment of all ratings may be required in our approach after some configuration changes such as those that modify the weighting of reputation subcategories. The calculation of rater credibility is not considered in the experiments. Although our algorithm unlike FIRE, takes into consideration the rater credibility value β it does not affect the scalability of the algorithm. This is because that the credibility value is calculated only once for each update regardless of the size of the available ratings.

In relation to complexity, the two algorithms differ in that the algorithm proposed in this thesis always has a constant number of arithmetic operations that is not affected by the number of ratings stored in the database. The formula that calculates the reputation in FIRE algorithm however, adds an extra multiplication operation for each individual rating. Each rating is multiplied by its weight and then summed together. In our algorithm, the reputation is multiplied by the accumulated weight stored from the previous calculation and the latest rating is multiplied by its weight. This results in that FIRE algorithm takes linear time. The experimental results seem to reflect this effect on time taken to update trustworthiness.

The delay in processing time has an effect on the trustworthiness evaluation because of the role of time in the computations. Figure 4.7 compares the trustworthiness evaluations when (A) our prediction algorithm is used and (B) with FIRE algorithm during the time interval between 100 and 200 msec. Although the trustworthiness values are close they are not exactly the same because of the processing delay in case (B).

4.4.3 Effect of Changes in Trustworthiness in Constructs

The trustworthiness of a CS may be affected differently by changes in trustworthiness of its components if they are part of constructs requiring different approaches for the calculation. For example, as illustrated in Figure 4.8 (A) a moderate decline in the reliability of a component service executed in a loop results in a significant decline. In this experiment s_1 is executed three times in each execution of the CS. Also note that in this case reliability values are more significant than cost of a component service because of the exponential effect of changes in reliability. Consequently, it has significant effect on the CS optimisations.

In the case of exclusive choice construct the trustworthiness of a CS is only partially affected by decline of trustworthiness of one of the construct services depending on its probability of execution (see Figure 4.8 (B)). All three component services have equal probability ($33\frac{1}{3}\%$).

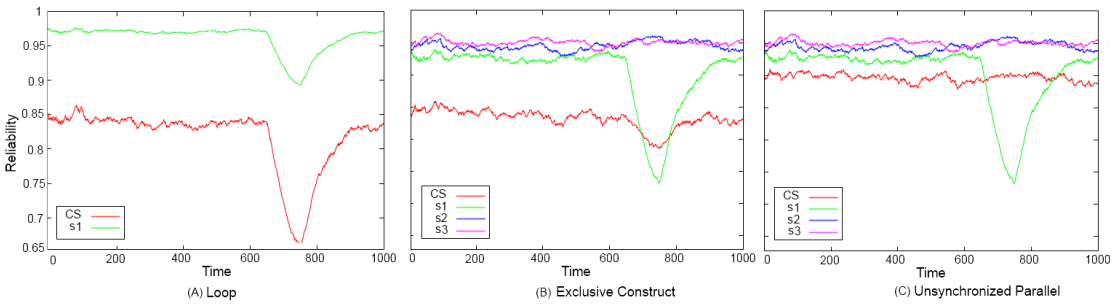


Figure 4.8: Effect of Reliability Drop on Constructs

In an unsychronized parallel construct $\{s_1, s_2, s_3\}$ as in Figure 4.8 (C). This decline does not affect the CS as the calculation method suggests as long as other services in the construct maintain their reliability. The reliability of the CS here is less than that of component services because of other sequence component services in the CS. Worthy of notice in this construct is that reputation and cost of a component play more significant role than its reliability because the reputation is the product of the reputation of all construct components and the cost is the sum of the costs all the component services.

4.5 Summary

CSPs need to be able to select trustworthy components for new compositions and respond swiftly to changed trustworthiness requirements and behaviour of existing CSs through adaptation. With the availability of alternative components providing the same functionality as those already integrated in a composition, CSPs can take advantage of this by replacing untrustworthy components.

This chapter presents an approach to monitoring and predicting the trustworthiness of composite services. The dynamic plugin-based trustworthiness module continuously monitors the adherence the services to their contracts and receives metrics relating to the reputation, QoS, security and other events. Trustworthiness attribute ratings are generated using a rules engine and stored in the module's trustworthiness ratings' store. The computation of trustworthiness depends on the construction of the composite service and the relative importance of components.

The chapter discusses the aggregation of trustworthiness attributes to collectively evaluate services and their constructs. Component services in a CS can be mandatory or optional components and may not all be equally important to the trustworthiness of the CS. Additionally, component services are executed in business processes which consist of different types of workflow constructs. The aggregation is based on the common structures of BPMN business processes. A custom Genetic Algorithm is used to optimise the composition of services based on their trustworthiness and cost.

Chapter 5

Security Metric Framework for Trustworthy Service Composition

5.1 Introduction

As discussed in earlier chapters, trust is a multifaceted concept that reflects extensible set of attributes of the services. However, security is a necessary step in establishing trust and maintaining service trustworthiness. In order to select and compose services that are most secure and trustworthy (see Scenario 2.5), there is a need for metrics to evaluate and rank those services in terms of their security attributes. The concept of attack surfaces [101, 102] is gaining more attention in the security research community. Measuring an attack surface in a software system involves identifying entry points and exit points in that system which are then used to determine the relevant potentially exploitable system resources.

In this chapter, we describe a novel framework for measuring attack surfaces in services environments, particularly, compositions of services. It describes service entry and exit points and classifies their security aspects and attributes. We consider the contribution of each of the resources that belong to a service based on the importance

and sensitivity of the resource. We also consider other factors that affect the effort needed to gain the required access and the extent of the resulting damage.

The computation of the attack surface metric depends on the types and the number of service resources and the level of compliance of the service to optimum values of applicable security aspects and attributes. We identify five quantifiable security aspects for resources in each entry and exit point in a service, which are *access privilege*, *access right*, *encapsulation*, *cryptography* and *validation*. The security attributes that belong to each aspect may vary depending on a range of considerations such as the type of service, its environment and implementation methods. However, this chapter describes a number of commonly required security attributes in atomic and composite services. Additionally, the chapter describes an approach to examining the security attributes of business processes based on common security problems and we relate these attributes to the attack surfaces of composite services.

The research contributions of this chapter include the application of the concept of attack surfaces to the service oriented computing. In particular, it contributes the composition of attack surfaces of atomic services together with the attack surfaces of their business processes. The chapter also contributes in taking a multifaceted view of the concept of attack surfaces and the classification of security aspects of services based on the protection methods and the exploitability of the service resources. Another contribution is the use of the Structured Assurance Case Model [31, 32] in measuring the security attributes of resources belonging to entry and exit points of components and their re-evaluation as components are incorporated into service compositions.

The chapter is organised as follows. Phases in optimising and computing the attack surface of a composite service are discussed in Section 5.2. Section 5.3 describes the structure of the attack surface of an atomic service and its computation. The section also details the security aspects and attributes of attack surfaces and the use of SACM in attribute evaluations. Issues related to the attack surfaces of composite services and their computation techniques are discussed in Section 5.4. A discussion and evaluation

of the framework are in Section 5.5. A summary of the chapter is described in Section 5.6.

5.2 Phases in Attack Surface Computation

The CS provider aims to provide a secure and trustworthy service with quantitative evaluation of its security as well as overall trustworthiness. In order to achieve this, the provider follows the phases outlined in Figure 5.1 which are detailed in the next sections. Building a secure business process requires the selection of secure component services and their encapsulation and secure integration. In addition to initial knowledge of component security metrics, the CSP has to measure the business process security metric as a whole. The provider requires to minimise the extension of the attack surface caused by the integration of the component services and the provision of the CS to the consumers as an integrated service. Composition of services can create new opportunities for attackers or introduce new security requirements. Examples of such requirements are constraints in relation to the access rights to component services or required process exception handling rules.

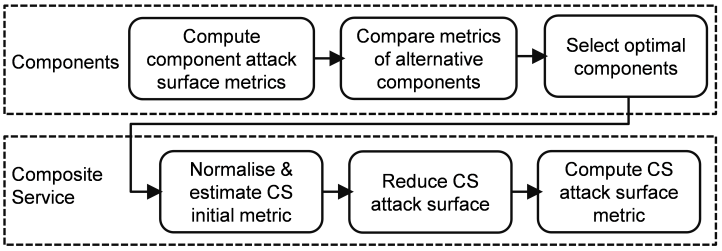


Figure 5.1: Phases in Optimisation and Computation of the Attack Surface of a Composite Service

5.3 Attack Surface of Component Service

Security researchers such as Bertino et al. [83] classify vulnerabilities of software components into two categories which are; first, software defects caused by design flaws or coding errors, and second, configuration errors due to unnecessary functions or access

control misconfigurations. Therefore, security issues can affect various elements of software that form either a route to the target, the actual target of an attack or their security relevant attributes. Consequently, those elements and their attributes constitute the software's attack surface.

For each service s_i that is available in an environment there is a set of entry and exit points which are based around the service operations specified in the service description as well as other interfaces of the service to the environment. Each point is associated with a set of exploitable resources with each resource having a security profile. The security profile specifies the security aspects of the resource. This classification of the security aspects and attributes of service resources is based on the protection methods that can affect the exploitability of resources. The following steps are required in order to determine the service attack surface.

1) Identify all entry and exit points of the service.

2) Determine the resources (i.e. operation and data items) associated with each entry/exit point.

3) For each resource, document its security profile in terms of its security aspects and attributes.

4) Determine values of resource security aspects and attributes using the software assurance case model.

5) Compute the damage potential to effort ratios for each entry and exit point based on the security aspects of its resources.

6) Compute the service's attack surface metric by aggregating damage potential to effort ratios of the service's entry and exit points.

The following subsections discuss each of the concepts to which we referred and describe the approach to achieving these steps.

5.3.1 Entry and Exit Points and Associated Resources

Figure 5.2 illustrates the categories of service entry and exit points. A service entry point is a service operation that has one of the following interactions:

- supported service interface (e.g. WSDL - Web Service Description Language) operations that can receive data from an interacting consumer or other service.
- service operations that execute external operations and receive data from other applications e.g. legacy applications.
- service operations that receive data from a file system or a data store e.g. LDAP or SQL database.

A service exit point, on the other hand, is a service operation that has one of the following interactions:

- supported service interface operations that can send data to interacting consumer or other service.
- service operations that execute external operations and send data to another application.
- service operations that send data to a file system or a data store.

Each entry/exit point h described above may have several resources associated to it. We categorize those resources as follows:

– *Operation (op):* a service operation is a key resource in an entry/exit point and its security profile significantly affects the security level of the service.

– *Data (d):* various types of data included in operation parameters, return types and persistent data can be associated with different levels of exploitability.

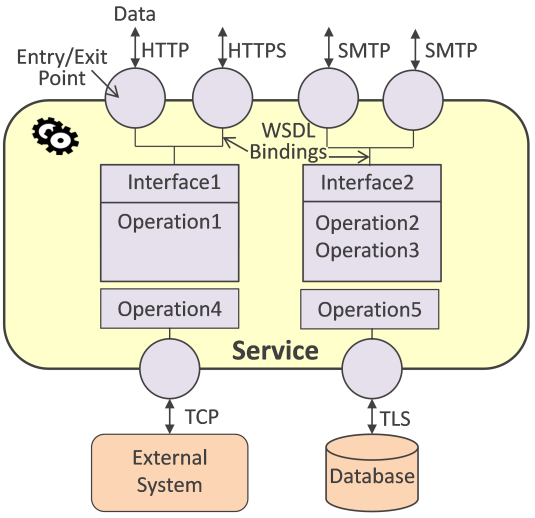


Figure 5.2: Example Service Entry/Exit Points

5.3.2 Resource Security Profile

Each potentially exploitable resource in an entry or exit point has a security profile. A profile consists of the resource's security aspects and attributes. The following security aspects are documented for each resource. The classification of the security profile of service resources into aspects and attributes is based on the protection methods that can affect the exploitability of the resources. The aim of the security aspects is to cover all security attributes of resources belonging to a service while every security aspect includes attributes that focus on certain application protection level. Some of the aspects are not applicable to all types of resources.

(*i*) *Privilege (G)*: Resources such as operations may have or can acquire multiple privilege levels. For example, an operation may access a database using certain excess privilege such as "root" allowing to bypass necessary checks when it can perform its required actions with less privilege.

(*ii*) *Access right (A)*: This determines the authentication required to gain different levels of access to a resource e.g. write, read, execute. An insecure resource may for example allow writing for unauthenticated users. The access rights can be role-based, attribute-based or a mixture of both. In order to assign scores to those

rights they are ranked according to the level of access and control they provide over the resource.

(*iii*) *Encapsulation (E)*: Encapsulation is software development technique that restricts access to operations and data items. A resource is more secure if it is internal rather than it has public access. Restriction of access to operations and data items, by setting them as non-public, protects from direct accessibility to the operations as well as protecting from other indirect security problems. For example, some unnecessarily public operations may result in exposed object construction and mutable objects which can have security implications. In addition to setting data programmatically to public access, other approaches can make them accessible such as serialisation. Therefore, there are existing approaches to restricting access to sensitive data in those cases as well e.g. exclusion from serialisation.

(*iv*) *Cryptography (C)*: This aspect is relevant to data. The application of encryption, use of encrypted channel, message encryption, key strength and proper key management affect potential attacks against a service's confidentiality. Digital signatures provide data integrity and non-repudiation. In WSDL-based services descriptions (Figure 5.2), each WSDL binding is associated with a protocol e.g. SMTP, HTTP, HTTPS. Since a service operation can have multiple bindings, more than one protocol may be associated with each entry/exit point. Other forms of entry/exit points, such as interfaces to backend systems, may also use one or more protocols. Therefore, a resource may have more than one level of security for the same aspect attribute (e.g. encrypted & unencrypted channels). In that case the principle of the weakest link is followed i.e. count the least secure. Cryptography applies also to logging data to ensure its confidentiality and integrity.

(*v*) *Validation (V)*: Validation controls are relevant to data and operations and they ensure proper detection and handling of security-related events and errors. This aspect can be examined under the following subcategories.

- *Data Validation*: Various levels of validation for XML schema and content enhance the security of services. The validation can be ranked according to; first, its exhaustiveness or degree of coverage and, second, the degree of protection provided by the recovery techniques. For example, common types of validations that protect against a variety of attacks include validations of XML syntax, data format and data size constraints. Each validation technique protects from a set of potential attacks e.g. constraining payload sizes provides protection of availability from Denial of Service (DoS) attacks.
- *Exception Handling*: This helps achieve robustness of the operations and protects from common vulnerabilities such as buffer overflow. Some research works exist on automatic testing of exception handling and recovery in applications such as the work of Marinescu and Candea [139] with limited research providing particular consideration of Web services and composition [140–142]. The degree of coverage of the exception handling techniques can be evaluated by first specifying the meaning of coverage. For example, coverage with respect of code text, control flow graph, class of errors, etc. Avižienis et al. [143] categorise fault and error handling and corrective techniques to a number of categories such as rollback, compensation and isolation. The usefulness and the degree of protection of the techniques are application-specific. From a security perspective, the main goal is that the operation fails into a secure state.
- *Logging*: Secure and reliable logging is also necessary for recording security incidents, policy violations and operational errors as well as auditing [144]. Message logging can be performed using a number of mechanisms such as input and output interception.

Figure 5.3 illustrates the resource categories for entry/exit points and their relevant security aspects and attributes. A service has multiple entry/exit points which in turn have associated resources (association labelled 'has resource'). Each resource (light purple box with '🔒' sign) is connected to a set of security aspects labelled with 'protected by'. Each aspect (light yellow box with '🔒' sign) for a resource is associated

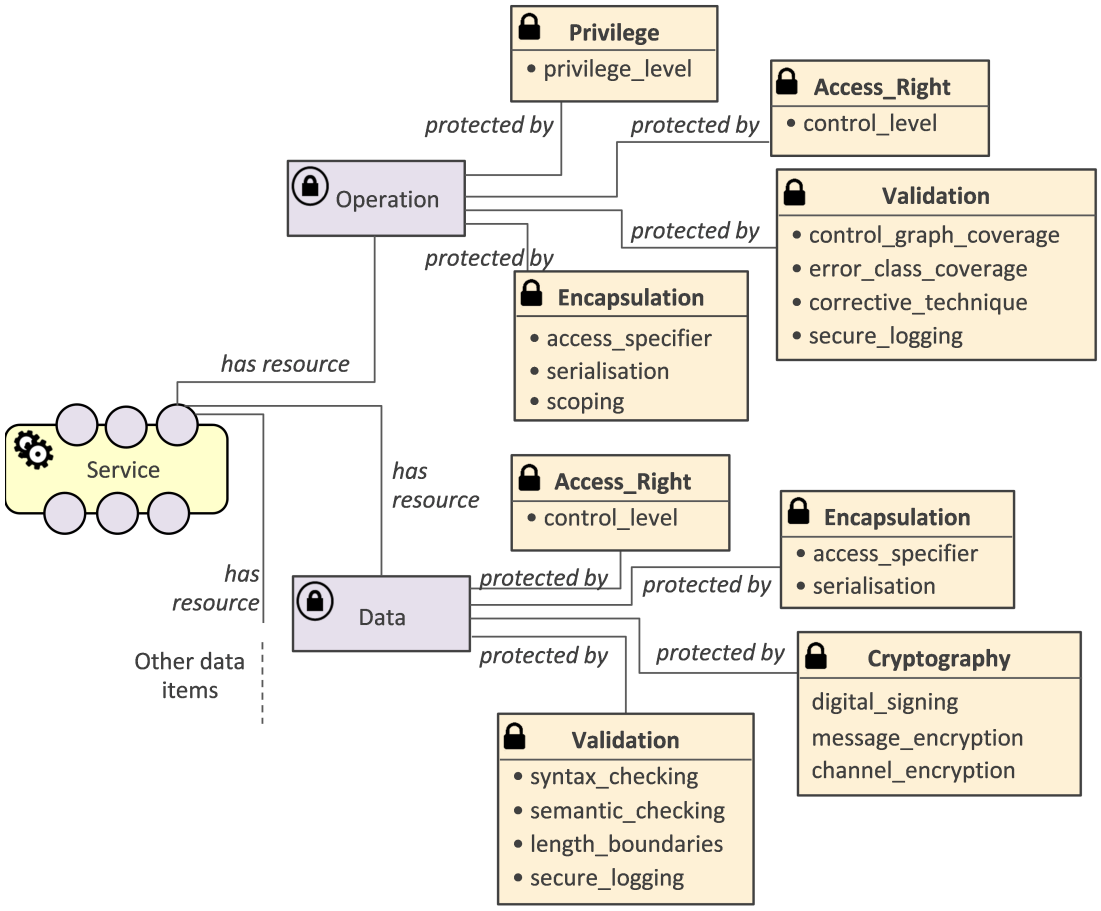
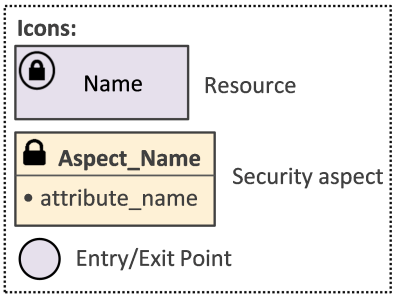


Figure 5.3: Service Entry/Exit Point Resource Types and Their Security Aspects



with a set of proposed common attributes to evaluate the aspect. These attributes can be extended or modified depending on the security-related characteristics of a service and its environment. Additionally, the attributes are also relevant to composite services as discussed in Section 5.4. Moreover, some attributes, such as *scope*, can be more relevant to composite than atomic services.

The aim of the security aspects is to cover all security attributes of resources belonging to a service while every security aspect focuses on certain application protection level. Each of the protection levels affects the exploitability of a resource belonging to a service. However, the implementation of the security aspects and attributes for a specific service instance may not in some cases properly include all required attributes for an aspect and this results in potential weakness in the defence layers and larger attack surface. Therefore, thorough implementation is essential in ensuring minimum exploitability of resources. For example, a DoS attack can occur due to buffer overflow vulnerability. However, if the validation security aspect includes code validation it should state the level of evidence that the code is tested and the extent of code coverage against classes of errors including buffer overflows. The same vulnerability (i.e. buffer overflow in this case) is normally protected through multiple security aspects and attributes as evaluated in Section 5.5.

5.3.3 Evaluation of Security Aspects Using SACM

As discussed in Subsection 3.4.2, approaches to modelling of software security attributes, vulnerabilities and threats depend on the purpose of the model such as identifying threats, their potential damage and prioritisation. The models may focus on analysing attacks, identifying countermeasures or risk mitigation.

In order to assign quantitative values to the security aspects and attributes described above we propose to use the Structured Assurance Case Model [31]. The model has been used widely in assuring system safety while some researchers such as Rhodes et al. [32] investigate its applicability to software. Boland et al. [103] describe an approach using SACM for the determination of software trustworthiness. We provide a brief description of the model and its applicability to the security aspect measurement.

The model consists of the following elements; *claims*, *subclaims*, *arguments* and *evidence* [145] as represented in Figure 5.4. Claims are propositions made to support satisfaction of security aspects and attributes. A claim may consist of supporting subclaims. Evidence is information or objective artefact used to support a claim or

subclaim. An argument contains the evidence and subclaims together with rules of inference to establish a claim. It should be noted that multiple subclaims, multiple evidence, etc represent logical conjunctions. That is the top claim is true if all subclaims are true. This is reflected in the computation of the value for the top claim (security aspect).

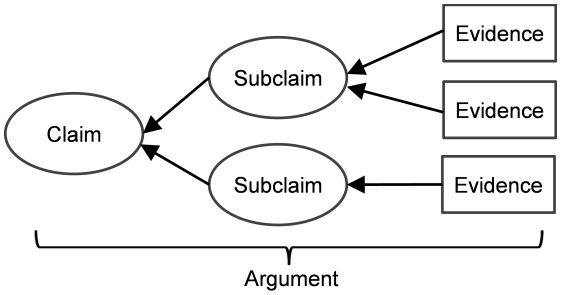


Figure 5.4: General Representation of the Structured Assurance Case Model

Figure 5.5 illustrates a summary of how the model can be used in determining the values for the security aspects and attributes using the validation aspect for an operation in the Loan service as an example (see Scenario in Section 2.5). The model is drawn using Adelard Assurance and Safety Case Environment (ASCE) [146].

In the figure the top oval is the top level claim Claim_1 regarding the Validation security aspect. Its subclaims (Nodes Claim_1.1 to Claim_1.4) in this example relate to the individual attributes as in Figure 5.3. Nodes Arg_1 to Arg_4 represent developed arguments that provide reasons why a claim should be accepted as true.

Nodes Ev_1 to Ev_4 are actual pieces of evidence. Based on each one, an *evidence strength* (*EvStrength*) is calculated in order to evaluate the so called *argument result* (*ArgResult*). Evidence supporting an argument should be as objective as possible and may take many forms such as static test results, simulation results, fault-tree analyses, modelling, certifications and human inspections. Security testing and verification techniques such as those described in the OWASP Application Security Verification Standard document [147] can be used as the basis of the evidences.

Evidence strength is a function of evidence strength factors such as sufficiency, accuracy and reproducibility. Multiple evidence items may contribute to the argument

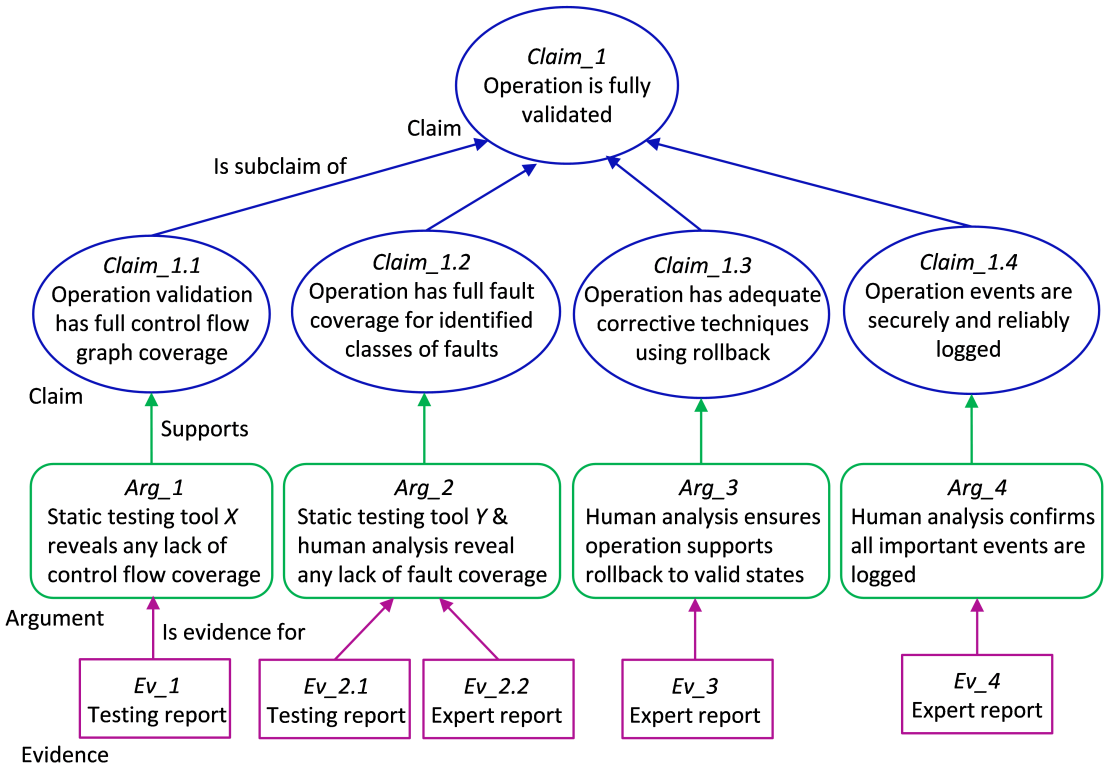


Figure 5.5: Example Structured Assurance Case for an Operation Validation Security Aspect

result such as Ev_2.1 and Ev_2.2 shown in the figure. For example, argument result may be calculated as a weighted average of supporting evidence strengths where ω is the weight of the corresponding item.

$$ArgResult\_1 = \frac{\omega_1 \times EvStrength\_1 + \omega_2 \times EvStrength\_2 + ...}{\omega_1 + \omega_2 + ...} \tag{5.1}$$

The value of a subclaim satisfaction is calculated as a function of related argument results. For example, this value can be calculated as a weighted average of supporting argument results.

The value of a higher level claim is a function of its subclaims. For example, a weighted product of subclaim satisfaction can be used to calculate a claim satisfaction. Similarly, a security aspect value (e.g. validation V_{op} for operation op) equals the value of top level claim satisfaction if only one exists, or otherwise a weighted product of multiple top level claims.

The top level claim or claims corresponds to a security aspect while subclaims represent security attributes. Claims and subclaims can be further decomposed into lower level subclaims with associated arguments and evidence as required in the application context. This requires calculating the value of a (higher level) subclaim using its subclaims. If Λ is a security attribute, the following formulas indicate how Λ can be calculated. The value N is the number of top level claims and M is the number of subclaims of one higher level claim that are at the same level. Note that a claim $Claim\_i.j$ can have recursively have further subclaims and can be calculated using the same Equation 5.3. Weights of subclaims are assigned independently from those of higher level claims. The sum of claims' weights at one level equals the number of claims at that level e.g. $\sum_{i=1}^{N} \omega_i = N$.

$$\Lambda = \prod_{i=1}^{N} Claim\_i^{\omega_i} \tag{5.2}$$

$$Claim\_i = \prod_{j=1}^{M} Claim\_i.j^{\omega_{i.j}} \tag{5.3}$$

In the case of the example given in Figure 5.5, the value V_{op} is calculated as follows.

$$V_{op} = Claim\_1 \tag{5.4}$$

$$Claim\_1 = Claim\_1.1^{\omega_{1.1}} \times Claim\_1.2^{\omega_{1.2}} \times Claim\_1.3^{\omega_{1.3}} \times Claim\_1.4^{\omega_{1.4}} \tag{5.5}$$

Further detailed description of the model is outside the scope of this thesis as our focus is on developing an approach for the use of the attack surface concept as a security metric for trustworthy service selection and compositions. The reader may refer to [32, 103, 145] for details on the methodology.

5.3.4 Resource Damage Potential to Effort Ratio

Resources in each entry/exit point contribute differently to the attack surface of a service. The contribution depends on three factors;

- the sensitivity and importance of the resource which are determined by internal or external characteristics of the resource such as its type, source or location,
- the effort required to access and attack the resource, and
- the potential damage or loss that can result from the attack. The damage is related to the resource characteristics as above or due to security aspects and attributes such as privileges.

The higher the damage potential or the lower the effort, the higher the contribution to the attack surface. Therefore, these factors determine what is called the *damage potential to effort ratio* of a resource and are estimated using the evaluation of the resource category and its security aspects. For example, to simplify the process of allocating quantitative values to data item resources they can be categorised based on data type, its location, source, etc. Values are assigned to each entry/exit point resource and its security aspects in order to compute the resources' damage potential to effort ratios.

Manadhata and Wing [102] use the privilege level to indicate the damage potential while the access right to the resource is used as the measure of the attack effort. They do not consider the other aspects of resource security described in this chapter although they mention the relevance of some of the aspects to the attack surface measurement. In software services, additional aspects described earlier including encapsulation, cryptography and validation are required to increase the attack effort. Encapsulation raises the attack efforts while cryptography and validation aspects both generally increase the efforts and reduce damage potential.

As a guideline on assigning numeric values to resources, Manadhata and Wing state that the values should be such that both the privilege values and the access rights values

affect the outcome of the attack surface measurements. They suggest choosing medium difference for the values of privilege levels and either a medium or a high difference for the values of access right levels.

Our proposed equation supports the consideration of encapsulation, validation and cryptographic scores as described in Subsection 5.3.3. We use *Weighted Product Model* from Multi-Criteria Decision Making [148] to balance the contribution of different security aspects in the attack surface metric. The variable ω_A refers to the weight of access rights, and so on for the weights of other aspects. For a security aspect with index k;

$$\omega_k \geq 0, \quad \sum_{k=1}^{l} \omega_k = l \tag{5.6}$$

where l is number of security aspects.

Equation (5.7) computes the damage potential to effort ratio $K_{h,i}$ for an entry/exit point h in a service s_i.

$$K_{h,i} = \frac{R_{op} \times {G_{op}}^{\omega_G}}{{A_{op}}^{\omega_A} \times {E_{op}}^{\omega_E} \times {V_{op}}^{\omega_V}} + \sum_{d=1}^{n} \frac{R_d}{{A_d}^{\omega_A} \times {E_d}^{\omega_E} \times {C_d}^{\omega_C} \times {V_d}^{\omega_V}} \tag{5.7}$$

where R_{op} is the entry/exit point's operation value as an attack surface resource and G_{op}, A_{op}, E_{op} and V_{op} are the assigned values for privilege, access right, encapsulation and validation respectively. Similarly, d refers to each of n data items in the entry/exit point and the variables correspond to the values of security aspects of the data items i.e. C_d refers to cryptography aspect value and so on for other aspects. Values allocated to a resource (R_{op} and R_d) depend on the factors described earlier in this subsection. Denominator variables e.g. A_{op}, must not be assigned zero value.

The attack surface metric value $\mathcal{S}_i$ for a service s_i is the sum of the resources' damage potential/effort ratios of all its m entry/exit points.

$$\mathcal{S}_i = \sum_{h=1}^{m} K_{h,i} \tag{5.8}$$

5.4 Attack Surface of Composite Service

5.4.1 Attack Surface Metric in Selection and Initial Estimation

The initial attack surface of a composite service $\mathcal{S}_{cs,init}$ is the sum of the damage potential/effort ratios (Equation 5.9) for all entry/exit points in its n selected components.

$$\mathcal{S}_{cs,init} = \sum_{i=1}^{n} \mathcal{S}_i \tag{5.9}$$

The selection of most trustworthy service composition (see scenario in Section 2.5) requires a comparable metric of security which allows for the aggregation of trustworthiness attributes as described in Chapter 4. Determining the maximum and minimum security metrics for a given service task is required in order to normalise the metrics for component services. It also helps to benchmark the security level of all component services and provide an initial estimation of the overall security of the composite service.

The maximum and minimum attack surface metrics ($\mathcal{S}_{i,max}$ and $\mathcal{S}_{i,min}$ respectively) are computed based on the expected types and quantity of resources in a service. The types and quantity of resources in turn depend on the nature of functionality, complexity and other characteristics of the service. The computation of the maximum and minimum metrics involves the same steps in determining the attack surface of a typical real world service. The maximum attack surface metric indicates the lowest security and the opposite for the minimum metric.

The minimum metric equals the attack surface metric (computed using Equations 5.7 and 5.8) when the highest values of the security aspects (except privilege level) are in place with the least applicable privilege levels and lowest possible number and contributions of entry/exit point resources. The opposite values of these variables are used to calculate the maximum metric. One of the main difficulties is the estimation of the minimum set of the needed resources for the service functionality in terms of operations

and data. Alternatively, the metric $\mathcal{S}_{i,min}$ may be calculated using the highest values of the security aspects (but lowest privilege level) of the same number and contributions of the resources of a selected service s_i.

Equation (5.10) shows how to determine the normalised metric value $\mathcal{S}_{i,norm}$ for a service s_i.

$$\mathcal{S}_{i,norm} = \frac{\mathcal{S}_i - \mathcal{S}_{i,min}}{\mathcal{S}_{i,max} - \mathcal{S}_{i,min}} \qquad \mathcal{S}_{i,max} \neq \mathcal{S}_{i,min} \tag{5.10}$$

5.4.2 Composition Effects on CS Attack Surface

As discussed above, component services must be secure to build a secure composite service because any insecure component can be used to subvert the whole process. In addition to ensuring security of component services, providers of composite services need to consider minimising the extension of the attack surface caused by integration of components and provision of the CS to the consumers as whole service. Secure composition can impose additional security requirements that aim to limit its attack surface. A composite service has its own entry/exit points in addition to those of its components. For example, in a BPMN-based process a service task is carried out by some kind of a service such as a Web service or an automated application. Therefore, it can be performed by a remote component service or a CSP's local application. Other task types including script tasks and business rule tasks may also create additional entry/exit points to a composite service.

Each of the entry/exit points has the two categories of resources described in Section 5.3 and the relevant security aspects and attributes of each resource. A composite service can reduce the attack surface through limiting access to component entry/exit points. As described by Gennari and Garlan [149], there are two ways to improve the attack surface metric for a given system; first, by reducing the number of externally

available resources and second, by reducing the benefits gained from exploiting a resource.

Both the specific requirements of a CS and changes to the attack surface of component services can affect claims, arguments and evidence in the SACM model used to evaluate security aspects of resources. This consequently affects the overall calculated attack surface of the integrated components. The following is the description of each of the security aspects in relation to composite services.

i) Privileges: Like other software systems, secure business processes need to enforce least privileges in accessing resources including the minimum privileges required to carry out a process task and only for the duration of that task.

ii) Access Rights: The composition may have new security requirements such as when there are required constraints in relation to the access rights to component services. Binding of Duty (BoD) and Separation of Duty (SoD) are common constraints in workflow security. In BoD the same user is required to perform two different activities while in SoD two different users must execute two distinct activities. Extensions to BPMN to support those constraints are proposed by researchers such as Brucker et al. [84] and Rodríguez et al. [85]. Bertino et al. [83] propose an authorization model called RBAC-WS-BPEL for BPEL business processes that aims to support the specification of such access constraints. These models can provide an extra shield against attack on component services and limit the extension of the attack surface caused by the composition.

iii) Encapsulation: A CS business process encapsulates the component services restricting external access and access from other components and providing a layer of protection. Resources from component services' entry/exit points including both operations and data items that are handled only inside the business process without direct involvement in interactions outside the process (i.e. with consumer, external system, data I/O) have reduced contribution to the attack surface from the CS perspective. For example, the data may be processed only internally;

received by the process from one component and then passed possibly with format or content changes to another component. However, the resources are not considered completely internal because of the distributed nature of the process interactions.

In addition, scoping elements (using enclosing <scope> tags in BPEL and *SubProcess* in BPMN) provide additional encapsulation through contextual grouping of tasks into subprocesses. In Figure 5.6 three component services (Rate, Credit History and Loan) from the composite service described in the scenario (Section 2.5) are grouped in a BPMN subprocess (white rectangle) that can also raise events. The *compensation* event for Loan service is internal to the subprocess which is signified using '◎'. The subprocess boundary events are *timer* event '◎' and *error* event '◎'. The activities external to the subprocess (e.g. "Additional Details") have no access to its variables. The end event '⊗' is called *cancel* event which is triggered in this case if allowed time is exceeded.

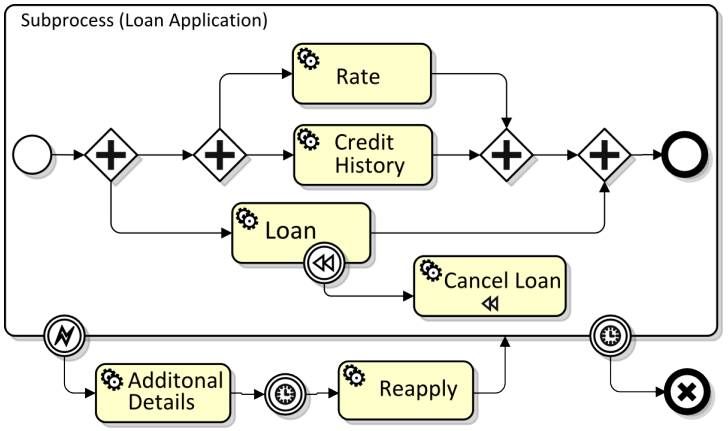


Figure 5.6: Encapsulation of Activities in a Subprocess

iv) Cryptography: A secure business process encrypts and signs messages between entities in the process so that only the recipient can decrypt them ensuring their integrity, authenticity, confidentiality and SoD. Robust cryptography techniques in a business process require compatible encryption algorithms and proper key management. In practice, performance issues are equally important requiring adequate cryptographic techniques that do not constrict efficient operation. Therefore, the maximum value assigned to this security aspect in any business process

in its attack surface computation does not necessarily correspond to the strongest possible encryption and signing mechanisms.

v) Validation: A composite service can provide its own validation rather than relying completely on data validation, exception handling and logging techniques supported by components. This consequently improves the validation coverage and/or the corrective techniques.

- *Data Validation*: Exchange of data between component services requires further data handling in a business process including validation, reformatting, transformation and storage. These procedures result in changes in the evaluation of security aspects of existing component data items and can impose additional data items, operations and entry/exit points.
- *Exception Handling*: Composition of services can create new opportunities for attackers and new vulnerabilities such as those resulting from deficiencies in exception handling within a business process. An exception can be thrown by a service runtime environment due to a certain condition in the process itself such as a join failure. An exception can also occur as a result of error conditions in the environment, network communication failures or other reasons. Those exceptions if not caught and handled properly may result in security breaches, for example, to the integrity or confidentiality of data. Therefore, a process requires further exception coverage and corrective techniques in addition to those implemented by its components.

  Exceptions are more likely in complex business processes. For that reason, component services and activities can be grouped into subprocesses to enhance the robustness of a composite service and improve the management of exception handling as in Figure 5.6 described earlier. In BPEL, this can be implemented using *scopes*. A scope (<scope>) allows to define handlers of exceptions as well as message and time-based events for individual activities or groups of activities in a workflow construct such as sequence or

parallel constructs. Exception handling techniques and their testing and verification in service compositions are also investigated by researchers such as Kuk and Kim [141] and Friedrich et al. [142].

- *Logging*: Like atomic services, secure and accessible logging is an important security attribute in business processes [8]. In addition to local component service logging, composite service logging allows to centrally trace security-related events relating to component services and the operation and interactions of the business process. Ideally, integration of distributed logs from various vendors allows a complete picture of events. However, interoperability issues are frequent with the absence of common implemented logging standards.

5.4.3 Computation of CS Attack Surface

Since some of the characteristics of service environments are that it is distributed and loosely-coupled, it may not be possible for a CS to impose restrictions on the quantity and security of the entry/exit points of component services. Therefore, a composite service inherits those entry/exit points. The initial evaluation of a CS attack surface described in Subsection 5.4.1 is only indicative of the actual attack surface of the developed composite service. The optimisation and measurement of a CS attack surface involves the following steps.

Step 1) Re-evaluate and reduce the attack surface for all components:

Each of the component services has associated entry/exit points with resources and security profiles as described in Section 5.3. The components attack surface could be further reduced using techniques such as BPEL security extensions, through modifying the component configurations, by communication interception or combination of these. The following are examples of the attack surface reductions for the pre-existing entry/exit points.

– moving component's operations from public to internal access, resulting in concealing the component's entry/exit points. Additionally, a component may include excess functionality. In such case, the disabling of such functions when applicable, for example by reconfiguration, can reduce the entry/exit points' attack surface or even remove some entry/exit points.

– providing more secure centralized data store for the composite service. Consequently, it reduces the attack surface caused by the security aspects of data resources. Since, this redistribution of data resources can modify the data categories (due to change of the location attribute as described in Subsection 5.3.2) it also can reduce the contribution of those resources.

As described earlier, composite service requirements and changes to components such as their modes of interaction can affect the claims, arguments and evidence in the SACM model used in evaluating security attributes. Therefore, it requires re-evaluating the affected attributes. The composition can also introduce new attributes such as scoping (Figure 5.3).

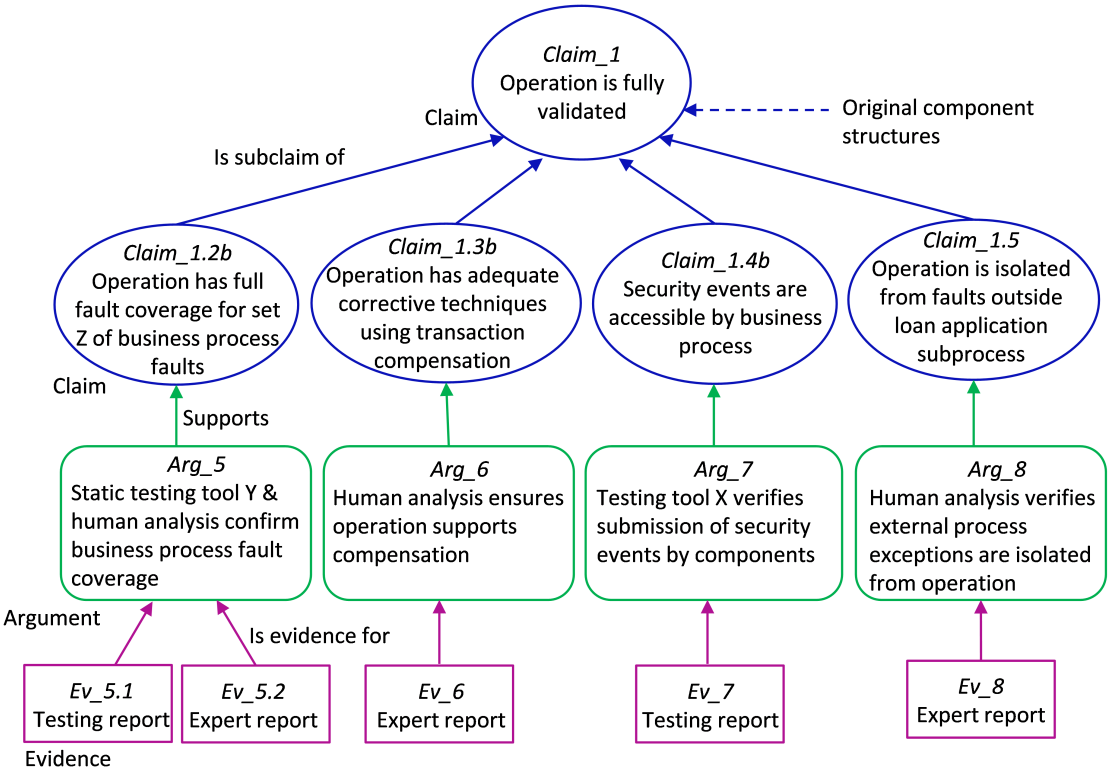


Figure 5.7: Example Structured Assurance Case for Validation Security Aspect of the Attack Surface in Updated Loan Component Operation

Figure 5.7 illustrates changes to the structured assurance case for the validation aspect of an operation in the Loan component described earlier in Figure 5.5 after consideration of the changes due to composition. Dotted blue arrow indicates unmodified original structures (Claim_1.1, Claim_1.2, etc) from Figure 5.5. Some claims complement existing ones to support the composition. For example, Claim_1.2b complements Claim_1.2 to ensure support for business process related exceptions, and similarly for other claims; Claim_1.3b (support for transaction compensation) and Claim_1.4b (support for log sharing). Other claims may be new such as Claim_1.5 that requires encapsulation through grouping of tasks using subprocesses and scopes (representing scoping attribute).

The updated damage/effort ratios for resources in entry/exit points of each component s_i are recalculated as in Equation (5.11). The acute accent indicates the modified security aspects and other variables based on requirements and changes performed in composite services as discussed above.

$$\acute{K}_{h,i} = \frac{\acute{R}_{op} \times \acute{G}_{op}^{\omega_G}}{\acute{A}_{op}^{\omega_A} \times \acute{E}_{op}^{\omega_E} \times \acute{V}_{op}^{\omega_V}} + \sum_{d=1}^{n} \frac{\acute{R}_d}{\acute{A}_d^{\omega_A} \times \acute{E}_d^{\omega_E} \times \acute{C}_d^{\omega_C} \times \acute{V}_d^{\omega_V}} \tag{5.11}$$

The attack surface metric value $\acute{\mathcal{S}}_i$ of each component s_i is then updated as below.

$$\acute{\mathcal{S}}_i = \sum_{h=1}^{m} \acute{K}_{h,i} \tag{5.12}$$

Step 2) Identify added entry/exit points of the composite service

A non-component entry/exit point in a composite service is an operation that exists outside the component services and is invoked as part of the business process logic. As with component services, a non-component entry/exit point in a composite service must have one or more of the types of interactions i.e. consumer, external system, data I/O. Examples of such entry/exit points that can be introduced in business processes are those associated with event handling, exception handling and business rule operations that support those interaction types.

Step 3) Evaluate security profile of resources associated with non-component entry/exit points:

The same approach of identifying resources and documenting their security profiles applies to the additional entry/exit points specific to the composite service. All security aspects are relevant to business process entry/exit points with adjusted considerations and priorities as described in Subsection 5.4.2. For example, the process is subject to a range of additional exceptions, and therefore, reduction of the attack surface requires both adequate coverage and corrective techniques for those exceptions (Figure 5.3). Likewise, further encapsulation using the process *scopes* discussed above provides an added protection by dividing the business process into independent subprocesses. In order to assign quantitative values to the security aspects, SACM model is also applied to the security aspects of the CS non-component resources.

Step 4) Aggregate the updated resource damage potential/effort ratios and compute the CS attack surface:

Each exploitable entry/exit point in the CS contributes to its attack surface. The damage potential/effort ratio K_j of an entry/exit point j in the composite service (other than those from components) is calculated based on the resource contribution and the security aspects as in the case of component resources. The overall CS attack surface $\mathcal{S}_{cs}$ is the sum of the damage potential/effort ratios (Equation 5.13) for all entry/exit points in the composite service including the updated attack surface measurement of entry/exit points of its components.

$$\mathcal{S}_{cs} = \sum_{i=1}^{n} \acute{\mathcal{S}}_i + \sum_{j=1}^{z} K_j \tag{5.13}$$

where n is the number of components in the composite service, z is the number of the non-component entry/exit points.

5.5 Discussion and Evaluation

We consider the attack surface of a service as a set of entry and exit points as proposed originally by Manadhata and Wing [102]. Each entry/exit point has security related

features that affect the protection and the exploitability of resources associated to that point. Our approach is focused on the attack surface of a service as opposed to the detection or prevention of specific vulnerabilities. However, the security aspects we propose also cover various types of known categories of vulnerabilities and threats. Classifications and listings of such vulnerabilities and threats are discussed by researchers such as Tsipenyuk and McGraw [150] and in threat repositories and software security projects as in Common Weakness Enumeration (CWE) [151] and OWASP [152] respectively.

Table 5.1: Relationship of OWASP 2013 Top Ten Security Risks [152] to Security Aspects of Service Attack Surfaces

OWASP Risk	**Relevance to Security Aspect**				
	G	**A**	**E**	**C**	**V**
A1 - Injection	++	++	++	++	+++
A2 - Broken Authentication and Session Management	+++	+++	++	+++	+++
A3 - Cross Site Scripting (XSS)	++	++	++	++	+++
A4 - Insecure Direct Object References	+++	+++	++	++	++
A5 - Security Misconfiguration	+++	+++	+++	+++	+++
A6 - Sensitive Data Exposure	+++	+++	++	+++	+++
A7 - Missing Function Level Access Control	+++	+++	+++	++	++
A8 - Cross-Site Request Forgery (CSRF)	+++	+++	++	++	+++
A9 - Using Components with Known Vulnerabilities	+++	+++	+++	+++	+++
A10 - Unvalidated Redirects and Forwards	++	++	++	++	+++

High protection against risk +++
Medium protection against risk ++

Table 5.1 indicates the relevance of the widely referenced OWASP 2013 top ten Web application security risks [152] to the attributes of the attack surface security aspects. Although the list refers to vulnerabilities or categories of vulnerabilities they claim that they are designed around organisational risks. The list is chosen because it focuses on Web applications, their code and development process. This makes the list fit well with the scope of our work. The list is referenced by well-known projects such as Common Weakness Enumeration (CWE) [151] and by security companies e.g. Core

Security [153] and Applicure [154] in relation to their security products. The document is also referenced by well-cited publications [e.g. 150, 155].

In the table, a risk is of high relevance if the corresponding security aspect attributes provide direct protection when the attributes have optimum values. Medium relevance indicates that the attributes provide significant but less defence or a second level protection. Examples of medium relevance are cases where encryption or encapsulation prevent access to resources even after an attack occurs. All security aspects have medium or high relevance to those risks. Our allocation of the relevance ratings is based on description of the risks and their mitigation and related defence mechanisms for Web services in general such as those described in OWASP documents [147, 152], CWE [151] and by Bertino et al. [83]. However, specific Web services may have individual priorities in relation to risks and mitigation methods.

The utilisation of SACM allows dependable measurement of individual security attributes through the claim-argument-evidence model. It also allows for traceable and reproducible results. The Structured Assurance Case Model has been widely used in complex systems such as avionics and shown to be an effective approach in assuring safety [32]. Claims satisfaction in our attack surface metric can use evidences from existing testing and verification methods such as those described in the OWASP Application Security Verification Standard document [147]. In addition to such static evidences, the model can incorporate operational evidences [105]. One main challenge is to use objective and reproducible evidences which may not always be entirely achievable due to the expected dependence on expert analysis reports to support some structured arguments.

Figure 5.8 illustrates the effect on the attack surface metric resulting from changes in the value of a security aspect of a data resource and changes of the aspect's weight, using the access right aspect A_d as an example. The values of security aspects can range between 0 and 1. The figure shows the effect in both cases where other security aspects (i.e. validation, encapsulation and cryptography) are of high ($V_d = E_d = C_d = 1$) and medium values ($V_d = E_d = C_d = 0.5$). Resource contribution value R_d depends on the

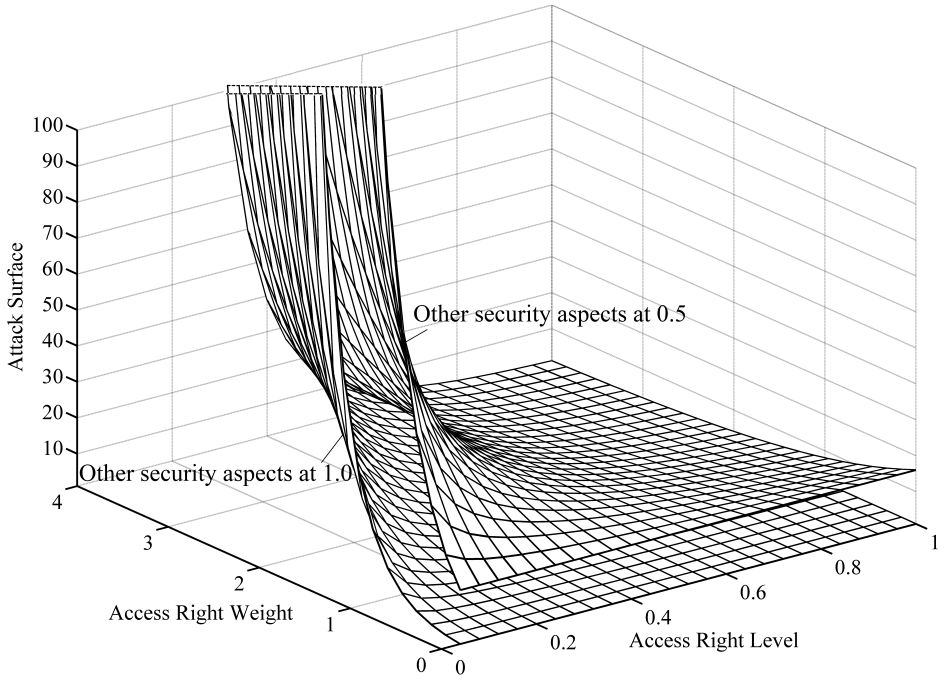


Figure 5.8: Change of Attack Surface Based on Values of Access Level and its Weight

significance and sensitivity of the resource. We set its value here at 1 which is the same as the highest value of the aggregated security aspects (i.e. 1). This means that the lowest potential damage/effort ratio is 1. The access right weight ranges from 0 (access right is completely disregarded) to 4 (access right is the only security aspect considered). The weight value for each security aspect is 1 for equally weighted aspects. As the figure shows, the resource attack surface score stays close to 1 (i.e. secure resource) when the access right (and other aspects) level score is high ($A_d \lesssim 1$), particularly if its weight is average or above (i.e. $\omega_A \geq 1$).

5.6 Summary

The chapter describes a framework to quantify security of services and their compositions as part of prediction of service trustworthiness. The chapter uses an approach based on the attack surface concept in quantifying security. It examines the features of attack surfaces in service environments including how an attack surface is composed in the case of service-based business processes. It describes service entry points and

exit points and classifies the security aspects and attributes of resources associated with these points. The computation of attack surfaces is based on contributions and security aspects of resources. The computed metric value depends on the types and quantity of the resources and the level of adherence of the service to the optimum values of the security attributes.

The evaluation of the resource security aspects builds on the Structured Assurance Case Model that is used to determine security aspect values through logical propositions called arguments. Those arguments aim to support claims using objective evidence data. Evidence supporting an argument may take many forms such as test and simulation results, modelling and human inspections.

The chapter also describes ranking of candidate component services based on their attack surfaces, evaluation of the attack surface metric of composite services and security issues relating to business process attack surfaces. Aggregated attack surface metrics for candidate services can be compared in order to select the likely most secure component service. A service security metric which represents the security attributes can be incorporated with other trustworthiness attributes of that service in order to select the most trustworthy component.

Once secure and trustworthy components for a composite service are selected the following step is to evaluate and reduce the overall attack surface of the business process. Security attributes of resources in the existing entry/exit points are re-evaluated using a new iteration of the structured assurance case according to the new requirements and updates of the service composition. In addition, non-component based entry/exit points in the business process are evaluated in the same way and added to the overall attack surface measurement of the composite service.

Chapter 6

Collaborative Trustworthiness Determination of Component Services

6.1 Introduction

In the scenario (Section 2.5) we described the need of CSPs to determine the values of component trustworthiness attributes based on those of existing composite services. The component attribute values are used by the CSPs to maintain and improve their composite service trustworthiness. The process may require service adaptation by replacing untrustworthy components. The determination of trustworthiness attribute values is also required for ranking and selection of components of new CSs.

A component service can simultaneously exist as part of business processes of multiple composite service providers and can jointly be invoked in those processes. Meanwhile, a CS may contain several such component services. Example of such services are the eCommerce services described in Section 2.5. The association between composite and component services can benefit CSPs and their consumers. The correlation between trustworthiness of CSs sharing some components allows determination

of trustworthiness of their components. The composite services can accordingly improve their trustworthiness during selection and afterwards via runtime adaptation by replacing untrustworthy components.

Service *reliability*, *reputation* and *response time* are some of the important attributes of trustworthy services [23, 24, 112]. The reliability of composite services depends on how reliable their components are. We measure reliability of a service as the percentage of its successful completion of executions. Reputation of a service is its overall value based on its consumer satisfaction ratings. Additionally, components of a composite service contribute differently to the service and their importance therefore varies. This consequently affects the component influence on the reputation of the CS. Response time is the length of time required to complete the execution of a service. Additionally, unexpected limitation of service *capacity* can cause delay or unreliability of a service [133] and consequently affect its trustworthiness.

In this chapter we describe novel framework and mechanisms for determination of those important trustworthiness attributes of components i.e. reliability, reputation, response time and capacity, based on monitoring and consumer reporting of the corresponding attributes of composite services. The distributed CSs jointly invoke distributed component services. CSPs set up contract agreement between each other to ensure fair collaboration and exchange of compiled monitoring and QoE data with each other to mutually maintain their trustworthiness and protect them from untrustworthy components.

The CSs can have varying degrees of similarity in terms of their components and they may have different process structure. In order to determine a trustworthiness attribute for a set of component services, we first try to find a system of equations in which the number of equations is equal to the number of variables. Each equation corresponds to the structure of the process of a particular composite service. The variables in those equations correspond to attributes of the components. We solve the linear or nonlinear system of equations to provide a unique solution. If equal number of equations and variables is not found we use the equations as constraints in an optimisation

problem that predicts the minimum trustworthiness of candidate components.

The chapter is organised as follows. Section 6.2 describes the framework and the trustworthiness determination procedure. Section 6.3 discusses the general mechanisms for the computation of component trustworthiness attributes. Section 6.4 describes determination of component reliabilities, reputations, response times and capacities respectively based on availability of equal number of equations and variables. Determination of those attributes in case of unequal number of equations and variables is discussed in Section 6.5. A summary is described in Section 6.6.

6.2 Trustworthiness Data Exchange

6.2.1 Exchange Framework

Our proposed framework for the exchange of trustworthiness data between CSPs is illustrated in Figure 6.1. The CSs share component services which are orchestrated by each CSP and executed by its *Execution Management* software module. The components of each composite service are indicated by the corresponding dotted coloured arrows. The composite services are also shown in Figure 6.4. Consumers can execute a composite service and submit QoE reports that indicate either success or failure of the execution and their satisfaction level. The CS is also monitored for its execution success and time by the Execution Management module. Each composite service has a *Trustworthiness Agent* that is responsible for exchanging trustworthiness data with corresponding agents and computing component trustworthiness. The circle connecting the agents exemplifies the communication channel to exchange the data. As demanded by the distributed and loosely coupled service environment, the mutual and fair collaboration and exchange of trustworthiness data between providers must be ensured through a contract-based agreement between providers. The details of the agreement is outside the scope of this thesis.

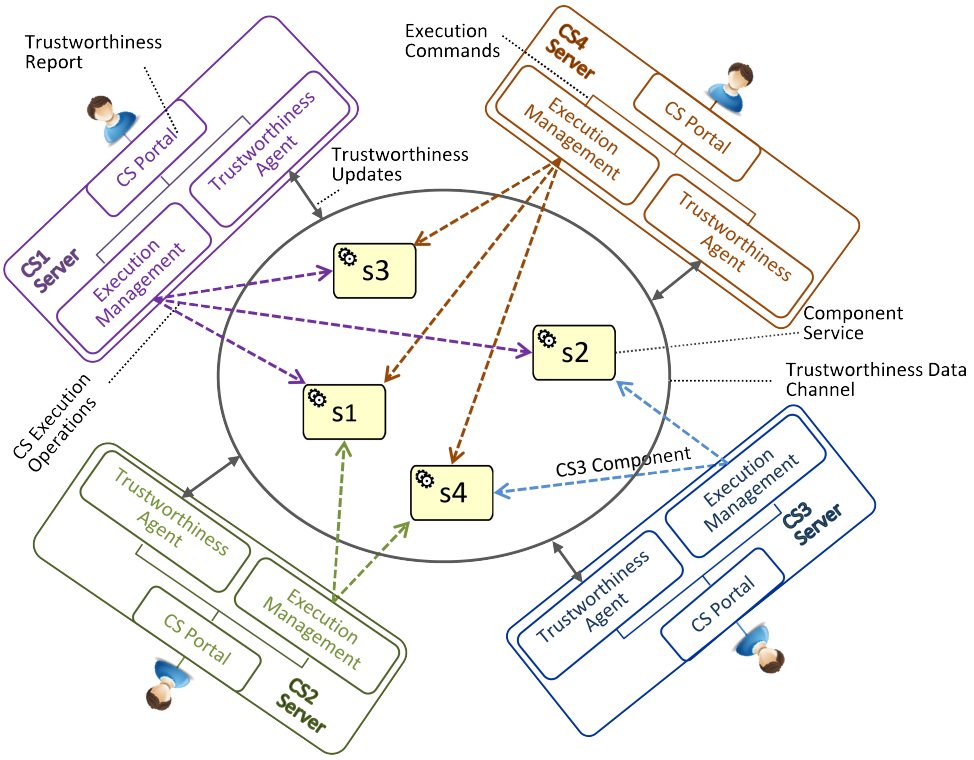


Figure 6.1: Trustworthiness Data Exchange Framework

6.2.2 Component Trustworthiness Determination Procedure

Each execution of a composite service either succeeds within or outside an appropriate response time or otherwise fails. The execution also provides variable satisfaction levels to consumers. Therefore, after each execution of a composite service, the following data are recorded; a reliability rating $R_i \in \{0, 1\}$, a satisfaction rating $0 \leq P_i \leq 1$ and response time T_i in msec, where i is the rating index. The reliability of a composite service is measured through the number of successful executions to the total executions l over a specified period of time ($1 \leq i \leq l$). For example, 50 failures out of 1000 executions in the specified period means that the CS reliability is 0.95. The failure indicates that a component service has failed. The values of the CS reputation and the response time are the average consumer satisfaction ratings and response times over that period respectively. The ages of the reliability and consumer ratings and of the response time can also be taken into consideration as described in Subsection 6.2.3. Since component services are shared between CSs we aim to use this bidirectional

association to determine the reliabilities, reputations, response times and capacity of components.

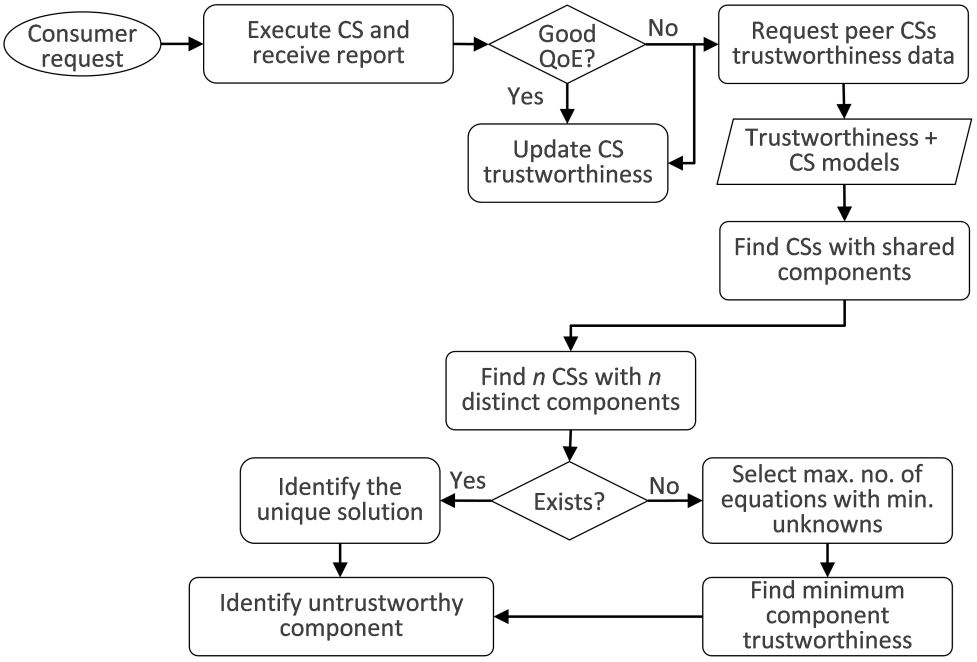


Figure 6.2: Procedure of Data Exchange and Determining Untrustworthy Component Service Performed by Local Trustworthiness Agent

Figure 6.2 describes the procedure for updating the trustworthiness of a composite service and its components after a consumer request. A failure, excessive response time or low satisfaction indicated by a consumer QoE report triggers collection of trustworthiness statistics by the local Trustworthiness Agent from peer composite service providers. The statistics are used to determine components reliabilities, response times and reputations and detect the component causing the failure, delay or bad reputation. The exchanged data includes BPMN models of the related CSs in XML. Based on the data, the Trustworthiness Agent checks if it can set up enough equations where each equation represents own or peer composite service (see Section 6.3).

The unknowns are the component reliabilities or other trustworthiness attributes. The number of equations should equal the number of unknowns in order to find a unique and accurate solution. If not, the Agent can solve the problem by the minimum trustworthiness for a component using optimisation as detailed in Subsection 6.3.3. In that

case, the Agent selects the maximum number of equations with the minimum number of unknowns in order to find to obtain a most accurate result.

6.2.3 Framework Scalability and Accuracy Discussion

Each local Trustworthiness Agent is responsible for the computation of the attributes of its provider's composite service. The effect of the complexity of the CSs on the computation is not significant because, first, each agent is concerned only with data from limited set of other CSs that have similar components and, second, the computation is carried out only when there is a need to update the trustworthiness. Examples of cases where trustworthiness update is needed are when there are negative consumer reports or when a provider plans a service adaptation or new composition. Therefore, the framework is scalable in relation to those aspects.

The framework also scales well with an increase in the number of consumers and the number of composite services. This is because the increase in the consumers helps improve the accuracy of the results due to the ability to rely on a wider range of consumers as well as filter those ratings that are considered unreliable. As described earlier individual ratings are processed locally which as a positive side-effect improves scalability. The increase in the number of CSs is also beneficial as it makes it easier to identify CSs with shared components.

The approach assumes either moderately consistent patterns of behaviour of component services over time or the use of most recent consumer ratings. In case of inconsistent patterns together with the use of ratings over longer time intervals the results of the attribute determination may not be reliable due to unreliable input that may not represent the current state of services. However, this can be overcome by the use of ageing of ratings over time which is described in Subsection 4.2.4.

An important related issue that has been studied by researchers is the credibility and honesty of consumer ratings and of sources of trustworthiness data such as collaborating providers. Examples of such research include the work by Walsh and Sirer [60], Malik

and Bouguettaya [7] and Hoffman et al. [56]. The main existing approaches in dealing with the problem are through majority rule and rating of consumers and data sources [7, 60]. However, the incentive for bad ratings in our case is low because composite service providers are not interested in high ratings from consumers but rather accurate ratings that reflect the performance of the service and its components. Those ratings are used indirectly to determine trustworthiness of components. Nonetheless, inaccurate ratings are possible for other reasons such as components promoting themselves or simply misjudged ratings. The filtering of bad consumer ratings and honesty of other providers are not the focus of this thesis. On the other hand, the computation approach has varying levels of tolerance to the inaccuracy of monitoring and consumer based ratings as discussed in the following sections. The tolerance depends on the extent of the feasible solutions and the rate of diversion of the blame on bad trustworthiness from one component towards another as a CS trustworthiness attribute value deviates.

6.3 General Approach to the Computation

A number of issues are considered when computing the trustworthiness attributes of CS components, these include:

- The characteristics of each trustworthiness attribute which determine the attribute aggregation technique.
- The structure of the CS process (see Section 2.4) which affect the linearity of the equation representing the composite service.
- The degree of similarity and distribution of joint components among collaborating CSs, which help provide more accurate predictions.
- The probability of execution of components in Exclusive Choice and in Multi-choice with Synchronised Merge constructs.

6.3.1 Computation Steps

The following describes the proposed steps that a provider takes towards computing the trustworthiness of its CS components. The next sections discuss each attribute separately and provide examples for different cases.

Step 1) Find equal number of equations and variables:

Assemble a set of equations whose size m is at least equal to the number of their variables z i.e. provider's CS components that have unknown values. We use $\bar{S}$ to indicate the set of components of all CSs with unknown attribute values, that is

$$\bar{S} = \bigcup_{j=1}^{m} S_j \tag{6.1}$$

where S_j is the set of components of a composite service CS_j and j is a numeric identifier for the composite service. In order to achieve equal equations and variables there must be $|\bar{S}|$ linear or nonlinear equations (i.e. $m = z = |\bar{S}|$). If this can be achieved using linear equations then unique solutions can be found and the computation process ends. If the equations are nonlinear unique solutions can be found for reliability and reputation but not for response time. These are detailed in Subsection 6.3.2. If equal number of equations and variables cannot be achieved, go to step (2).

Step 2) Consider only a subset of components:

Assemble a set of equations whose size is not equal to the number of CS components but equals the number of the equations' variables and includes a subset of the CS components. The equation for our provider's CS is not included in that set of equations i.e. $S_j \not\subset \bar{S}$ but $S_j \cap \bar{S} \neq \emptyset$. The considered components may also include components that are not in the provider's CS. This step also provides a unique solution that includes the values for some of the CS components. If not all equations are linear, then the problem is solved as a nonlinear system as in Subsection 6.3.2.

If this step is achieved and some of the CS components become known, return to step (1) to compute the values for the remaining unknown CS components. Otherwise, go to step (3).

Step 3) Find a local solution instead:

If we cannot achieve equal equations and variables ($z > m$), then assemble best possible set of equations which is where there would be the least number of variables and maximum number of equations and $z \approx m$. Using optimisation find the worst possible trustworthiness attribute values for the components as discussed in Subsection 6.3.3. The approach is most suitable in finding out the worst predictable trustworthiness for an individual component or a construct of components forming part of a CS such as in service selection.

6.3.2 Case of Equal Number of Equations and Variables

If all equations are linear, the problem is solved using a linear solution for more accuracy and speed. The reliability r_{cs_j} of composite service CS_j with n Sequence components can be calculated as product of that of the components as in Equation (6.2) from Chapter 4.

$$r_{cs_j} = \prod_{i=1}^{n} r_i \tag{6.2}$$

Consequently, component reliability can be computed as follows.

$$\sum_{i=1}^{n} \log(r_i) = \log(r_{cs_j}) \tag{6.3}$$

where r_{cs_j} is reliability of a composite service CS_j. The formula also applies to other constructs where the reliability of the construct is a product of that of its components such as Synchronised Parallel.

The boolean function $g(s_i, CS_j)$ is a function that indicates whether s_i is a component of CS_j. If S_j is the set of components of CS_j then.

$$g(s_i, CS_j) = \begin{cases} 1 & if\ s_i \in S_j \\ 0 & if\ s_i \notin S_j \end{cases} \tag{6.4}$$

The reliability of the components can then be calculated using matrix multiplication as in the following formula.

$$\begin{pmatrix} g(s_1, CS_1) & g(s_2, CS_1) & \cdots & g(s_z, CS_1) \\ g(s_1, CS_2) & g(s_2, CS_2) & \cdots & g(s_z, CS_2) \\ \vdots & \vdots & \ddots & \vdots \\ g(s_1, CS_m) & g(s_2, CS_m) & \cdots & g(s_z, CS_m) \end{pmatrix} \begin{pmatrix} log(r_{s_1}) \\ log(r_{s_2}) \\ \vdots \\ log(r_{s_z}) \end{pmatrix} = \begin{pmatrix} log(r_{cs_1}) \\ log(r_{cs_2}) \\ \vdots \\ log(r_{cs_m}) \end{pmatrix} \tag{6.5}$$

The aggregation equation for the reputation of a composite service with n components and not containing Exclusive Choice or Multi-Choice with Synchronised Merge constructs can be obtained as follows.

$$p_{cs_j} = \prod_{i=1}^{n} p_i^{\omega_{i,j}} \tag{6.6}$$

where $\omega_{i,j}$ is the importance weight of the component s_i. By taking this aggregation technique we are assuming that the reputations of the components of a CS are interdependent and that the reputation of a CS is influenced by the importance of each component as well as by its reputation.

Accordingly, the reputation of components executed in those constructs, can be obtained using the following.

$$\sum_{i=1}^{n} \omega_{i,j} \cdot \log(p_i) = \log(p_{cs_j}) \tag{6.7}$$

where p_{cs_j} is reputation value of the composite service from which we aim to determine the reputation of the components.

If the function $\acute{g}(s_i, CS_j)$ is a function whose value is the weight of the component s_i when it is a constituent of CS_j or zero otherwise.

$$\acute{g}(s_i, CS_j) = \begin{cases} \omega_{i,j} & if\ s_i \in S_j \\ 0 & if\ s_i \notin S_j \end{cases} \tag{6.8}$$

Then the component reputations are computed as in Equation (6.5) using reputation values and $\acute{g}(s_i, CS_j)$ in place of $g(s_i, CS_j)$.

Similarly, the response times of Sequence components can be computed using the following formula. This is because the response time of a Sequence-only CS is the sum of that of its components.

$$\begin{pmatrix} g(s_1, CS_1) & g(s_2, CS_1) & \cdots & g(s_z, CS_1) \\ g(s_1, CS_2) & g(s_2, CS_2) & \cdots & g(s_z, CS_2) \\ \vdots & \vdots & \ddots & \vdots \\ g(s_1, CS_m) & g(s_2, CS_m) & \cdots & g(s_z, CS_m) \end{pmatrix} \begin{pmatrix} t_{s_1} \\ t_{s_2} \\ \vdots \\ t_{s_z} \end{pmatrix} = \begin{pmatrix} t_{cs_1} \\ t_{cs_2} \\ \vdots \\ t_{cs_m} \end{pmatrix} \tag{6.9}$$

Alternatively, if the equations for an attribute are nonlinear the problem is solved as a nonlinear system. In such cases, if $\bar{S}$ is the set of all variables (i.e. components of all CSs) then we must have $|\bar{S}|$ nonlinear equations. For instance, to calculate component reliabilities, the equation for CS_j can be represented as follows.

$$r_{cs_j} = f(R_j) \tag{6.10}$$

where f is a nonlinear function that is defined by the structure of CS_j, S_j is the set of components of CS_j (i.e. $S_j \subset \bar{S}$) and R_j is the set of the corresponding reliabilities of components in S_j.

6.3.3 Case of Unequal Number of Equations and Variables

The following describes the proposed steps that a provider takes towards determining the trustworthiness of set of components in a CS construct as part of the process of selecting the most trustworthy construct. The steps are illustrated in Figure 6.3. The later sections discuss each attribute separately and provide examples for different cases.

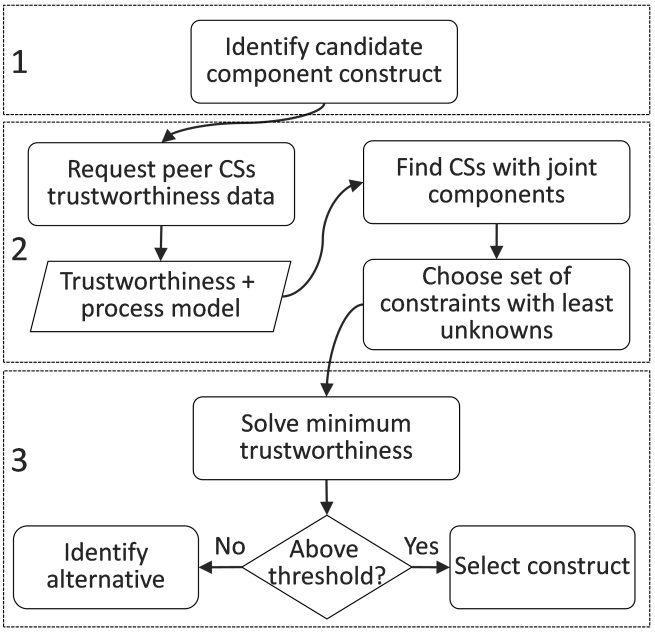


Figure 6.3: Procedure for Selection of Trustworthy Construct Components Performed by a CSP

Step 1) Consider a potential set of components for selection:

A CSP chooses candidate components that may be selected for a construct of a composite service on condition that their computed trustworthiness is above a threshold set in advance. The construct components can be a subset of all CS components. The type of the construct, e.g. Sequence, will define the objective function of the optimisation problem.

Step 2) Identify other CSs that use those components:

Identify a set of composite services from peer CSPs. The CSs must contain the chosen components and have known CS trustworthiness attribute values. Trustworthiness attribute values and CS process model (e.g. BPMN in XML format) are required for each of the CSs.

Each composite service defines an optimisation equality constraint whose variables are its components' attributes. The constraints can be linear or nonlinear. The considered CSs may also include other components that are not in the provider's CS. For most accurate results, the number of constraints should be close to the number of variables.

Step 3) Find the lowest predicted values for the trustworthiness attributes:

Optimisation is used to find the lowest predicted value of the attribute. If the values of all attributes e.g. reliability and reputation, are below thresholds the provider considers an alternative set of components. Otherwise, those components are selected. Another approach to the selection decision is to use a collective trustworthiness evaluation. A trustworthy construct may include components that are generally trustworthy but their attribute values vary. Therefore, the consideration of trustworthiness attributes individually and setting their thresholds can exclude some more trustworthy components or constructs than those selected. A common evaluation of trustworthiness can have advantages as it weighs all attributes together in the selection decision as described in Section 4.3.

Taking reliability as an example attribute with Sequence as the candidate construct, if the candidate components are a subset of all z components in the constraints, the problem for the subset (construct) reliability is a minimisation problem defined as follows.

$$\min \prod_{k \in z} r_k \tag{6.11}$$

subject to

$$f_j(r_i) = \alpha_j \quad \forall j = 1, ..., m; \ \forall i = 1, ..., z$$

$$0 \leq r_i \leq 1$$

where r_k is the reliability of component s_k, function $f_j(r_i)$ is a linear or nonlinear constraint based on a composite service CS_j with known reliability value α_j and r_i is each variable representing the reliability of a component in the constraints. This optimisation problem is subject to equality constraints that each represents a composite service. The value m is the number of CSs and their corresponding constraints. The boundaries for the values is as follows $0 \leq r_i \leq 1$.

The objective function depends on the type of construct the components are considered for. For example, the reliability of a Sequence construct can be calculated as the product of that of the components as in Equation (4.4). Therefore, the objective function is $\prod_{k \in z} r_k$ in Equation (6.11).

6.4 Attribute Determination Using Equal Number of Equations and Variables

6.4.1 Reliability

This subsection examines the approach described in the previous section in relation to reliability when equal number of equations and variables can be found. It also provides examples and testing. Consider the four simple composite services illustrated in Figure 6.4 as an example which share multiple components indicated by their numbers. All the CSs in the figure consist of only Sequence components.

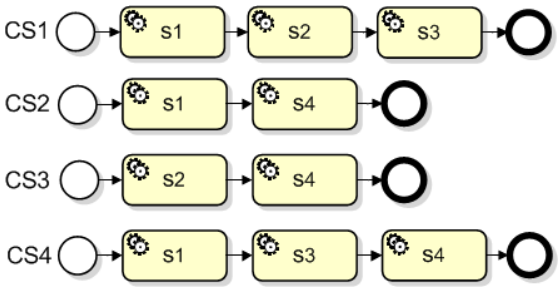


Figure 6.4: Sequence CSs Sharing Multiple Components

The following are the equations that determine the reliability of each CS in the figure based on that of its components.

$$\begin{aligned} r_{cs_1} &= r_{s_1} \cdot r_{s_2} \cdot r_{s_3} \\ r_{cs_2} &= r_{s_1} \cdot r_{s_4} \\ r_{cs_3} &= r_{s_2} \cdot r_{s_4} \\ r_{cs_4} &= r_{s_1} \cdot r_{s_3} \cdot r_{s_4} \end{aligned} \tag{6.12}$$

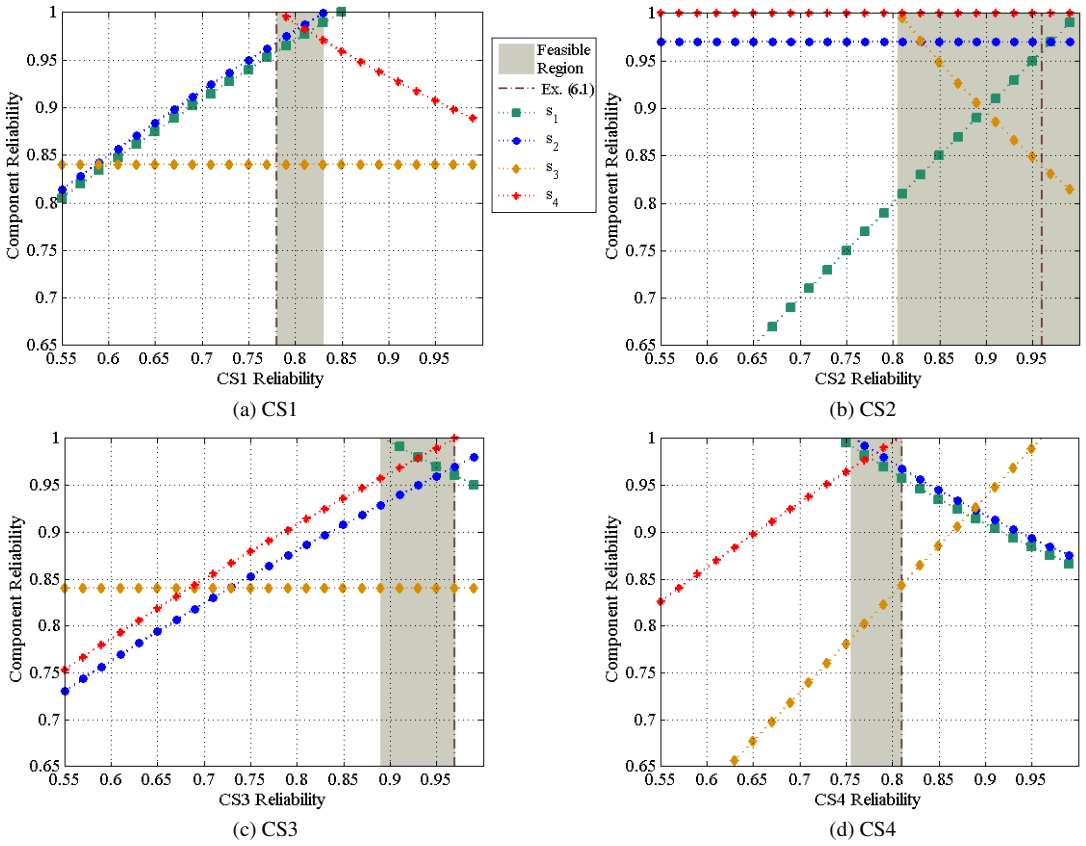
Example 6.1. Suppose that according to the collected reliability ratings, we have the following reliability values for CS1 to CS4 respectively; {0.78, 0.96, 0.97, 0.80}. Now we need to know the reliability of each of the joint components.

In the case of CS1 to CS4 we have four equations and four variables (Equation 6.12). Therefore, there is one solution to the problem. Accordingly, we can calculate the reliabilities of the components using Equation (6.5).

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} log(r_{s_1}) \\ log(r_{s_2}) \\ log(r_{s_3}) \\ log(r_{s_4}) \end{pmatrix} = \begin{pmatrix} log(0.78) \\ log(0.96) \\ log(0.97) \\ log(0.80) \end{pmatrix} \tag{6.13}$$

This results in the following values {0.96, 0.97, 0.84, 1.0} for the component reliabilities r_{s_1} to r_{s_4} respectively.

Figure 6.5 shows the feasible range of solutions to the problem when the reliability value for each of the CSs changes individually between 0.55 and 1.0, while the remaining composite services are fixed at values stated in Example (6.1). The shaded area is where the computed results are feasible i.e. all component reliabilities are between 0 and 1.0. The size and location of the feasible solution region depends on the subset of components in the changing CS and the distribution of those components in the remaining CSs. As the reliability value of the CS changes the net effect of multiplication of the reliabilities of its components has to result in an increase. Therefore, at least one component has to increase while other components would either increase, stay constant or decrease to a lesser extent. Meanwhile, the changes in the components' reliability values has to satisfy other equations (i.e. corresponding CSs) where the CS reliability is constant. Consequently, the resulting degree of change of components' reliability



(a) CS1 (b) CS2

(c) CS3 (d) CS4

Figure 6.5: Changes in Results of Component Reliability Determined from CS Reliabilities (Linear Solution)

values proportionately affects the size of the feasible solution region. The figure indicates that the untrustworthy component may or may not be affected by the change but that component is detected even when CS reliabilities have a slight range of error. The dash-dot line indicates the location of the solution to Example (6.1).

The variations of the case discussed above include composite services with other types of constructs and the degree of sharing and distribution of common components between CSs. Since the calculation of reliability in CSs containing Parallel and Unordered Sequence constructs is the same as an ordered Sequence-only composite service, the linear solution described above applies to such cases as well.

CS5 to CS8 in Figure 6.6 are more complex containing Unsynchronised Parallel (CS5), Exclusive Choice (CS6) and Multi-choice with Synchronised Merge (CS7) constructs. The service structure complexity requires more elaborate procedure to calculate component trustworthiness values. The reliabilities of the composite services in Figure 6.6 can be represented with a system of nonlinear equations which are based on construct aggregation equations discussed in Chapter 4.

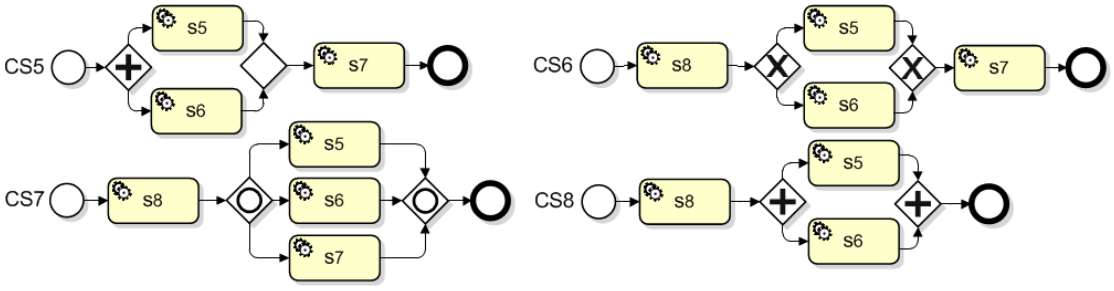


Figure 6.6: More Complex CSs Sharing Multiple Components

We assume that there is equal percentage of execution of components in Exclusive Choice and the percentage of Multi-choice with Synchronised Merge subsets is 30%, 30% and 40% for the combinations of components $\{s_5, s_6\}$, $\{s_6, s_7\}$ and $\{s_5, s_6, s_7\}$ respectively.

Accordingly, Equation (6.14) describes the system of nonlinear equations for CS5 to CS8.

$$\begin{aligned} r_{cs_5} &= (1 - (1 - r_{s_5})(1 - r_{s_6}))\, r_{s_7} \\ r_{cs_6} &= r_{s_8}(0.5\, r_{s_5} + 0.5\, r_{s_6})\, r_{s_7} \\ r_{cs_7} &= r_{s_8}(0.3\, r_{s_5} \cdot r_{s_6} + 0.3\, r_{s_6} \cdot r_{s_7} + 0.4\, r_{s_5} \cdot r_{s_6} \cdot r_{s_7}) \\ r_{cs_8} &= r_{s_5} \cdot r_{s_6} \cdot r_{s_8} \end{aligned} \tag{6.14}$$

The equations can be simplified to the following.

$$\begin{aligned} r_{cs_5} &= r_{s_5} \cdot r_{s_7} + r_{s_6} \cdot r_{s_7} - r_{s_5} \cdot r_{s_6} \cdot r_{s_7} \\ r_{cs_6} &= 0.5\, r_{s_5} \cdot r_{s_7} \cdot r_{s_8} + 0.5\, r_{s_6} \cdot r_{s_7} \cdot r_{s_8} \\ r_{cs_7} &= 0.3\, r_{s_5} \cdot r_{s_6} \cdot r_{s_8} + 0.3\, r_{s_6} \cdot r_{s_7} \cdot r_{s_8} + 0.4\, r_{s_5} \cdot r_{s_6} \cdot r_{s_7} \cdot r_{s_8} \\ r_{cs_8} &= r_{s_5} \cdot r_{s_6} \cdot r_{s_8} \end{aligned} \tag{6.15}$$

Because we have four equations and four unknowns there is only one solution to this non-linear system of equations.

Example 6.2. The following values are aggregated from the reliability ratings of CS5, CS6, CS7 and CS8 respectively; {0.92, 0.75, 0.68, 0.62}.

Using Matlab, we can use the nonlinear system solver; *fsolve* which by default uses the *trust-region method* for square systems (i.e. number of equations equals number of variables) or *Levenberg-Marquardt method* otherwise. The algorithm starting points for the component reliabilities are set at 0.5 for each variable. The function gives the correct solution which is {0.67, 0.92, 0.94, 0.99} for r_{s_5} to r_{s_8}. The solution indicates low trustworthiness of the component s_5.

Figure 6.7 shows the changes in the computed values of component reliabilities when the reliability of each of CS5 to CS8 individually changes between values 0.55 and 1.0 while the remaining composite services are fixed at values stated in Example (6.2). The resulting changes of component reliabilities indicate that the component to blame for the low trustworthiness may in some cases shift from one component to another as a result of a moderate change of a CS reliability. This is particularly evident for

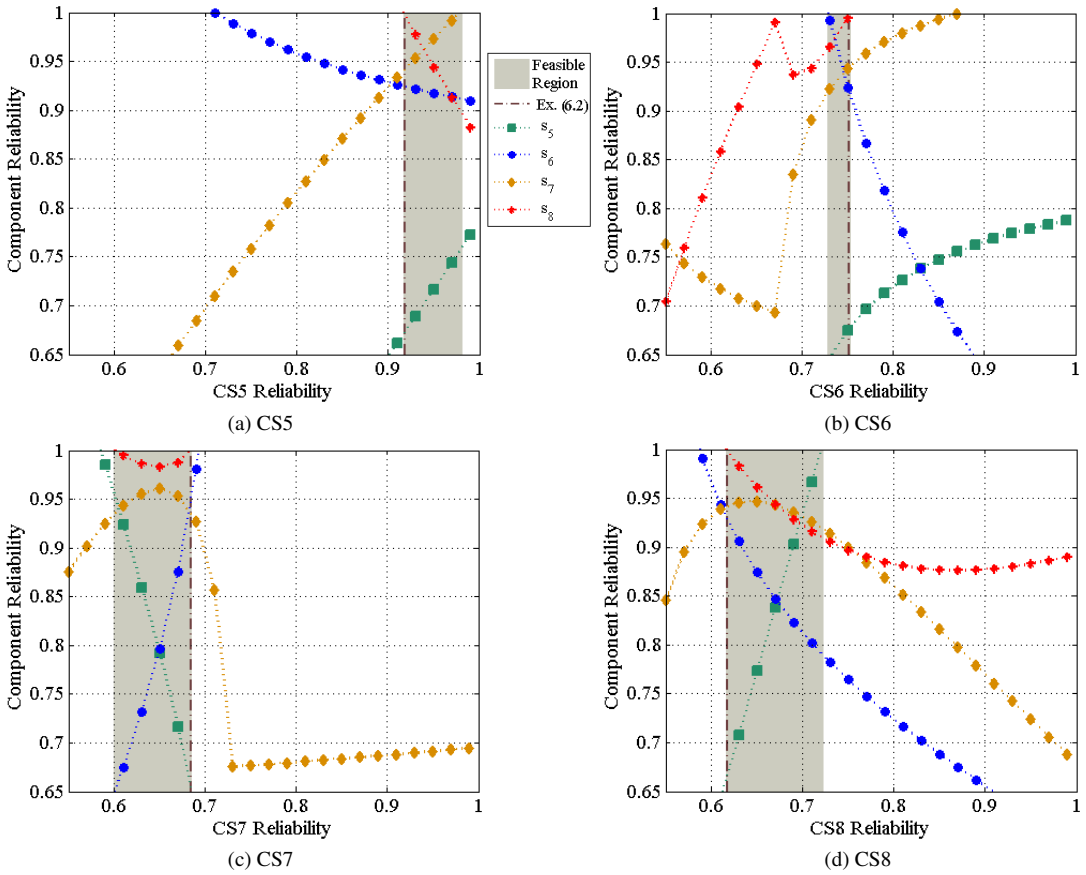


Figure 6.7: Changes in Results of Component Reliability Determined from Reliabilities of More Complex CSs

s_5 and s_6 in the case of CS7 reliability as in sub-figure 6.7 (c). In other cases component reliabilities don't change dramatically within the feasible solution space even with moderate changes in a CS reliability (e.g. subfigure 6.7 (a)). This indicates that the tolerance to CS reliability evaluation errors could vary from one case to another. The dash-dot line indicates the solution to Example (6.2) where all component's reliability values crossing the line are equal to the example solution. The line appears on the side of the feasible range because r_{s_8} is close to 1 in the example which is the end of the feasible range.

The 3D view in Figure 6.8 shows the feasible solutions to component reliabilities when both CS5 and CS6 reliabilities change. Only a limited range of values from CS5 and CS6 can provide solutions to the component reliabilities as some values do not allow possible solutions. Out of the possible solutions there is a subset that are feasible where resulting component reliabilities are between 0 and 1.

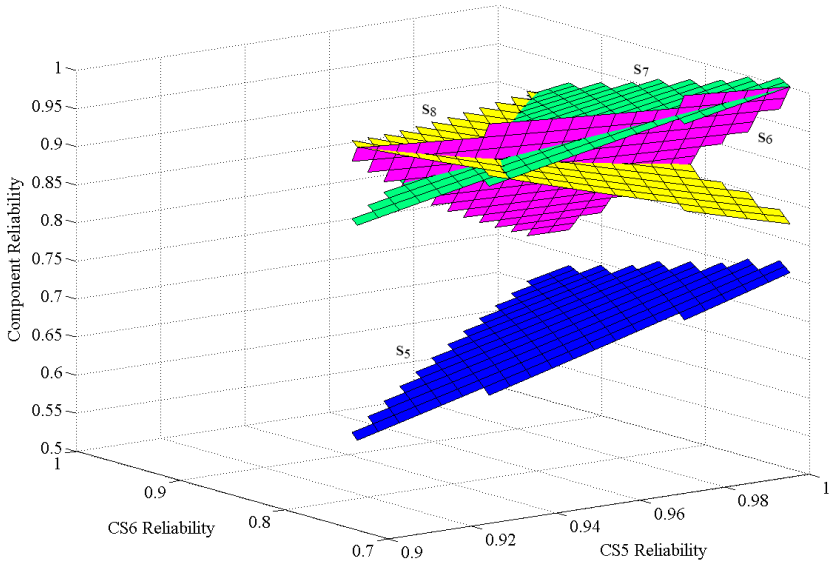


Figure 6.8: Feasible Results of Component Reliability as CS5 and CS6 Reliability Changes

6.4.2 Reputation

Calculation of reputation of components follows a similar approach to that of reliability but with consideration of the weights of components in each composite service and the difference in the aggregation formulas for some constructs. This approach is based on our assumption that the reputations of the components of a CS are interdependent and that the reputation of a CS is influenced by the importance of each component as well as its reputation. The reputation aggregation formulas are described in Section 4.2.3. The reputations of CS components can be computed from that of the CS using matrices as described in Section 6.3.

Example 6.3. Suppose that according to the consumer reports, we have the following reputation values for CS1 to CS4 (Figure 6.4) respectively; {0.71, 0.81, 0.92, 0.77}. The importance of each component (ω_i) in each CS is displayed in the leftmost matrix in Equation (6.16) where each row and column represents a composite service (CS1 to CS4) and a component (s_1 to s_4) respectively. We need to know the reputation of each of the joint components.

In this case we have the system of equations (based on Equation 6.7) and four variables. Therefore, there is one solution to the problem. Accordingly, we can calculate

the reputations of the components using Equation (6.16) as described in Section 6.3.

$$\begin{pmatrix} 1.1 & 0.9 & 0.7 & 0 \\ 1.1 & 0 & 0 & 0.9 \\ 0 & 0.7 & 0 & 1.3 \\ 1.0 & 0 & 0.6 & 1.4 \end{pmatrix} \begin{pmatrix} log(p_{s_1}) \\ log(p_{s_2}) \\ log(p_{s_3}) \\ log(p_{s_4}) \end{pmatrix} = \begin{pmatrix} log(0.71) \\ log(0.81) \\ log(0.92) \\ log(0.77) \end{pmatrix} \tag{6.16}$$

The resulting component reputations are $\{0.84, 0.92, 0.90, 0.98\}$.

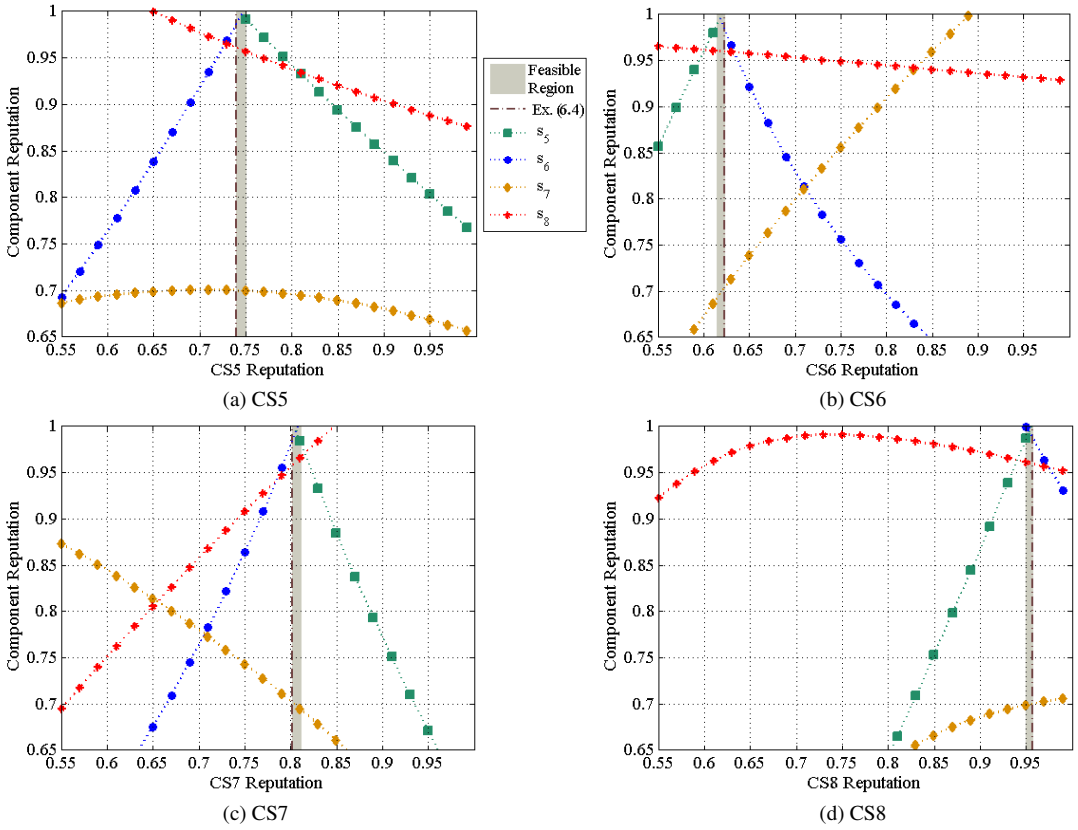
In case for more complex constructs, a more elaborate procedure is required. The reputation in case of both Synchronized and Unsynchronized Parallel is aggregated using Equation (6.6) as their components are always executed. For composite services with Exclusive Choice constructs and Multi-Choice with Synchronised Merge constructs (as in Figure 6.6) the component reputations are represented using nonlinear equations similar to the approach described for reliability. For example, the reputation p_θ of Exclusive Choice construct components can be aggregated using Equation (4.8). Therefore, the reputation of CS6 when s_5 and s_6 each has 50% chance of being executed in all CS6 executions can be represented using Equation (6.17). As a linear solution is not possible in this case, the problem is solved as a nonlinear system of equations.

$$p_{cs_6} = (p_{s_8})^{\omega_{s_8,cs_6}}(0.5\ (p_{s_5})^{\omega_{s_5,cs_6}} + 0.5\ (p_{s_5})^{\omega_{s_6,cs_6}})(p_{s_7})^{\omega_{s_7,cs_6}} \tag{6.17}$$

Example 6.4. Based on consumer reports, we obtained the following reputation values for CS5 to CS8 respectively; $\{0.74, 0.62, 0.80, 0.96\}$. The importance of each component (ω_i) in each CS is according to the matrix below where each row and column represents a composite service (CS5 to CS8) and a component (s_5 to s_8) respectively and the element value is based in Equation (6.8).

$$\begin{pmatrix} 0.9 & 1.3 & 0.8 & 0 \\ 0.7 & 1.0 & 1.2 & 1.1 \\ 1.0 & 0.8 & 0.6 & 1.6 \\ 1.2 & 0.9 & 0 & 0.9 \end{pmatrix}$$

Using the system of nonlinear equations representing the CSs and Matlab's *fsolve* function, the resulting reputations of the components s_5 to s_8 are as follows {1.0, 0.99, 0.70, 0.96} successively which is a unique and accurate solution.



(a) CS5 (b) CS6

(c) CS7 (d) CS8

Figure 6.9: Changes in Results of Component Reputation Determined from CS Reputations

Figure 6.9 shows the changes in the computed values of component reputations when the reputation of each of CS5 to CS8 individually changes between values 0.55 and 1.0 while remaining composite services don't change. The shaded area is where the computed results are feasible i.e. component reputations are between 0 and 1.0. The dash-dot line indicates the solution to Example (6.4) where each component's reputation values crossing the line in each subfigure must be equal. The line appears on the side of the feasible region because r_{s_5} is equal to 1.0 in the example which is the end of the feasible region. The feasible region in case of reputation is smaller than that in reliability because of more constraints on the solution as a result of importance weights. Figure 6.10 illustrates the wider feasible solution region when CS5 reputation is changing and importance weights of components are equal or not considered.

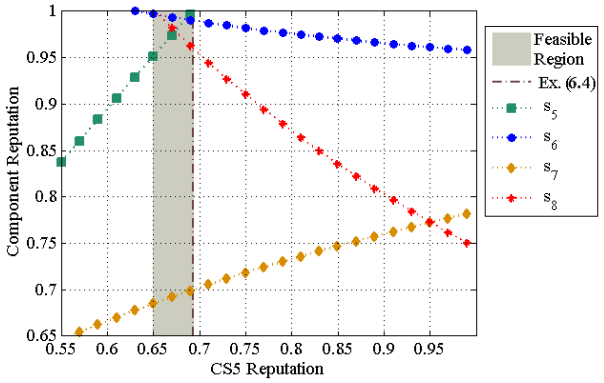


Figure 6.10: Changes in Results of Component Reputation Determined from CS Reputations without Importance Weights

6.4.3 Response Time

Response times of components can be determined from that of sequence composite services such as those in Figure 6.4 as in the example given below. A delay in a CS execution may be a result of a delay in one or more components. Component response times naturally vary depending on the characteristics of the service but the aim is to detect which component exceeds its advertised or agreed response time.

Example 6.5. The response times of the composite service CS1 to CS4 in milliseconds are {200, 280, 270, 330}. The accurate and unique solution for the response times of the components i.e.{80, 70, 50, 200} can be obtained based on Equation (6.9).

In the case of more complex composite services, the equations aggregating the response times may be nonlinear depending on the construct types. For example, in Synchronised Parallel constructs the components are executed in parallel but the next construct cannot commence its execution until all parallel components are complete. Therefore, the construct response time t_θ equals the longest of the response times of its components. On the other hand, in Unsynchronised Parallel the minimum of the components response times is the construct response time because the next construct starts when the first executing component in the construct completes. Hence, the response times of the CSs in Figure 6.6 can be represented as in the set of equations below.

$$
\begin{aligned}
t_{cs_5} &= \min(t_{s_5}, t_{s_6}) + t_{s_7} \\
t_{cs_6} &= t_{s_8} + 0.5\ t_{s_5} + 0.5\ t_{s_6} + t_{s_7} \\
t_{cs_7} &= t_{s_8} + 0.3\ \max(t_{s_5}, t_{s_6}) + 0.3\ \max(t_{s_6}, t_{s_7}) \\
&\qquad +0.4\ \max(t_{s_5}, t_{s_6}, t_{s_7}) \\
t_{cs_8} &= t_{s_8} + \max(t_{s_5}, t_{s_6})
\end{aligned}
\tag{6.18}
$$

The system of equations can be solved using nonlinear system solvers to identify the component response times. The solution to this problem can be easier and more accurate if the response times of some of the components are already known, for example, inferred from other composite services with joint components using linear problem solution as described in Example (6.5). The solvers of this nonlinear system of equations finds a local solution only. A guessed starting point for each variable is given which should be as close to the expected solution as possible. The guessed starting point may be set at the approximate component response times, for example, based on results of other calculations. There is usually multiple solutions although the resulting estimated component response times could be close to each other.

6.4.4 Capacity

One possible cause of the delay in execution is that the capacity of a component is exceeded. This in turn could be the result of the CSP exceeding agreed capacity usage or that the advertised component capacity is inaccurate. Approximate component usage at each particular time could be estimated based on the resulting response times and the component response time aggregation techniques. Response times calculated using a nonlinear solution may not be reliable because it only provides a local solution. The component usage can then be compared to the advertised capacity in the cases of unexpected delays. Algorithm 6.1 illustrates a procedure for estimating the usage u_i of a component service s_i at a point in time t_x.

```
for each CS_execution do
  if (t_{cs_j,end} > t_x & t_{cs_j,start} < t_x) then
    t_{i,start} = t_{cs_j,start} + t_{pre,i}
    t_{i,end} = t_{i,start} + t_i
    if (t_{i,end} > t_x & t_{i,start} < t_x) then
      u_i ← u_i + 1
    endif
  endif
endfor
```

Algorithm 6.1: Procedure for Estimating Usage of Component Service

First, the component execution start $t_{i,start}$ and end $t_{i,end}$ time points are estimated for each instance of a composite service execution spanning t_x. In such a CS execution, t_x must be between its start time point $t_{cs_j,start}$ and end time point $t_{cs_j,end}$. The component start and end time points are determined based on the CS execution start time point $t_{cs_j,start}$ and the aggregated preceding component response times $t_{pre,i}$. The preceding response times are determined according to the response time aggregation techniques described earlier. Then we determine whether the execution instance was ongoing during that point in time. If so we increment the usage u_i at t_x.

6.5 Attribute Determination Using Unequal Number of Equations and Variables

6.5.1 Reliability

This subsection examines the approach described in Subsection 6.3.3 in relation to reliability when it is not possible to find equal number of equations and variables. The subsection also provides examples and experiments. Consider the three simple composite services, illustrated in Figure 6.11, as an example which share multiple components each indicated by its number. All the CSs in the figure consist of only Sequence components.

The reliability r_{cs_j} of composite service CS_j with n Sequence components can be calculated as product of that of the components as in Equation (6.2). The formula also

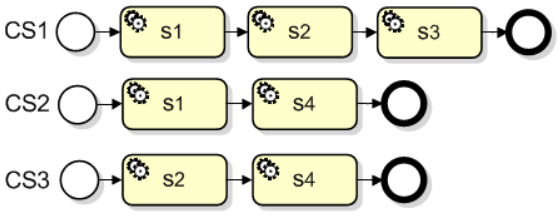


Figure 6.11: Sequence CSs Sharing Multiple Components

applies to other constructs where the reliability of the construct is a product of that of its components such as Synchronised Parallel. Therefore, if the value of r_{cs_j} is known ($r_{cs_j} = \alpha_j$), the following are the constraints that determine the reliability of each CS in Figure 6.11 based on that of its components.

$$\begin{aligned} r_{s_1} \cdot r_{s_2} \cdot r_{s_3} &= \alpha_1 \\ r_{s_1} \cdot r_{s_4} &= \alpha_2 \\ r_{s_2} \cdot r_{s_4} &= \alpha_3 \end{aligned} \tag{6.19}$$

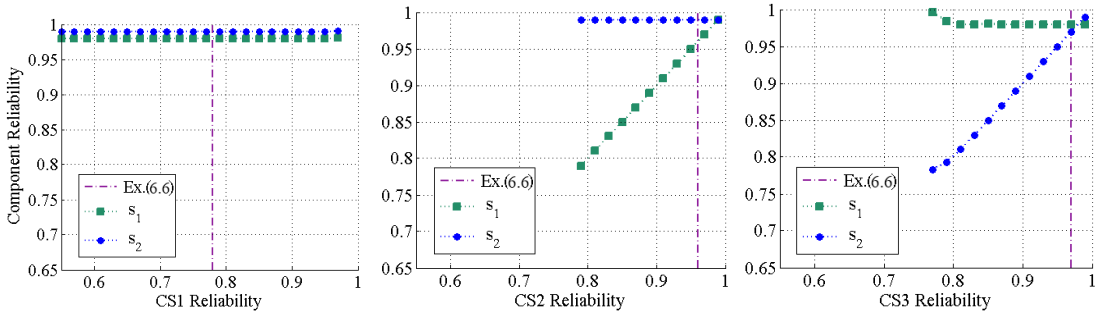
If a provider chooses the two components s_1 and s_2, the constraints in Equation (6.19) become the equality constraints for the minimisation problem described in Subsection 6.3.3. The objective function and constraints can also be linearised based on Equation (6.2) as follows.

$$\log(r_{cs_j}) = \sum_{i=1}^{n} \log(r_i) \tag{6.20}$$

Example 6.6. Suppose that in order to select components s_1 and s_2 for a Sequence construct in a CS, a provider requires that their aggregated reliability must be above 0.95. According to the collected reliability ratings, the following are reliability values for CS1 to CS3 (Figure 6.11) respectively; {0.78, 0.96, 0.97}.

We need to know the minimum possible aggregated reliability of s_1 and s_2 based on this data. In the case of CS1 to CS3 we have three constraints and four variables (Equation 6.19). Accordingly, we determine the minimum reliabilities of the components using the procedure described above and Matlab solver *fmincon* which supports finding the minimum of a constrained linear or nonlinear multivariable function. The resulting reliabilities are {0.98, 0.99} for the component reliabilities r_{s_1} and r_{s_2} respectively which give an aggregated reliability above 0.95.

Figure 6.12 shows the feasible range of solutions to the problem when the reliability value for each of the CSs changes individually between 0.55 and 1.0 while the remaining CSs are fixed at values stated in Example (6.6). The feasible values are when all component reliabilities are between 0 and 1.0 (lower and upper boundaries). The size and location of the feasible solution region depends on which subset of components the CS has and the distribution of those components in the remaining CSs. As the reliability value of the CS increases, the net effect of multiplication of the reliabilities of its components has to result in an increase. Therefore, at least one component has to increase while each of the other components would either increase, stay constant or decrease to a lesser extent. Meanwhile, changes in the components' reliability values have to satisfy other constraints (i.e. corresponding CSs) where the CS reliability is constant. Consequently, the resulting degree of change of components' reliability values proportionately affects the size of the feasible solution region. The figure indicates that the components (including the untrustworthy) may or may not be affected by the change but that approximate minimum is determined even when the reliability measurements of the CSs have a slight range of error. The dash-dot line indicates the location of the solution to Example (6.6).



(a) CS1 (b) CS2 (c) CS3

Figure 6.12: Changes in the Minimisation Solution of Components Reliability Determined from Composite Service Reliabilities (Linear Constraints)

Variations of the case discussed above include CSs with other types of constructs. Since the calculation of reliability in CSs containing Parallel constructs is the same as an ordered Sequence-only CS, the linear solution described above applies to such cases as well. CS4 to CS6 in Figure 6.13 are more complex containing Unsynchronised Parallel (CS4), Exclusive Choice (CS5) and Multi-choice with Synchronised Merge (CS6) constructs. Service structure complexity leads to more complex nonlinear constraints to determine a candidate construct's trustworthiness value. The reliabilities of the composite services in Figure 6.13 can be represented with a system of nonlinear constraints which are based on construct aggregation equations described in Subsection 4.2.3.

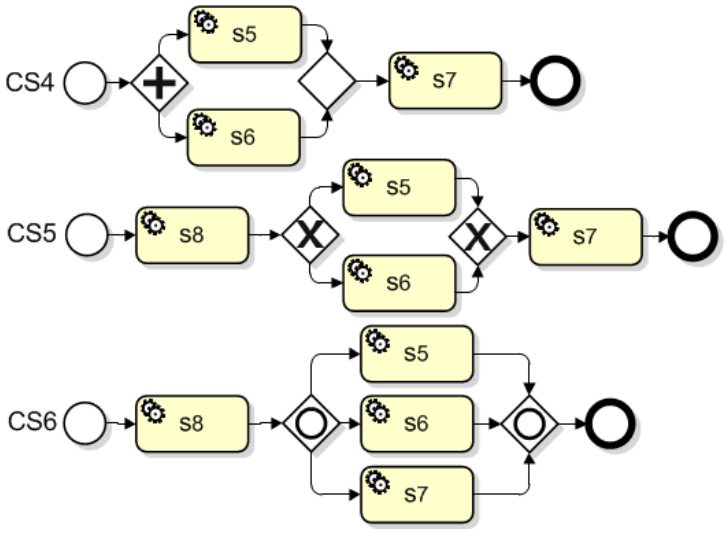


Figure 6.13: More Complex CSs using Multiple Joint Components

We assume that there is equal percentage of execution of components in Exclusive Choice and the percentage of Multi-choice with Synchronised Merge subsets is 30%, 30% and 40% for the combinations of components $\{s_5, s_6\}$, $\{s_6, s_7\}$ and $\{s_5, s_6, s_7\}$ respectively. Accordingly, Equation (6.21) describes the system of nonlinear constraints based on CS4 to CS6.

$$\begin{aligned} &(1-(1-r_{s_5})(1-r_{s_6}))\, r_{s_7} = \alpha_4 \\ &r_{s_8}(0.5\, r_{s_5} + 0.5\, r_{s_6})\, r_{s_7} = \alpha_5 \\ &r_{s_8}(0.3\, r_{s_5} \cdot r_{s_6} + 0.3\, r_{s_6} \cdot r_{s_7} + 0.4\, r_{s_5} \cdot r_{s_6} \cdot r_{s_7}) = \alpha_6 \end{aligned} \tag{6.21}$$

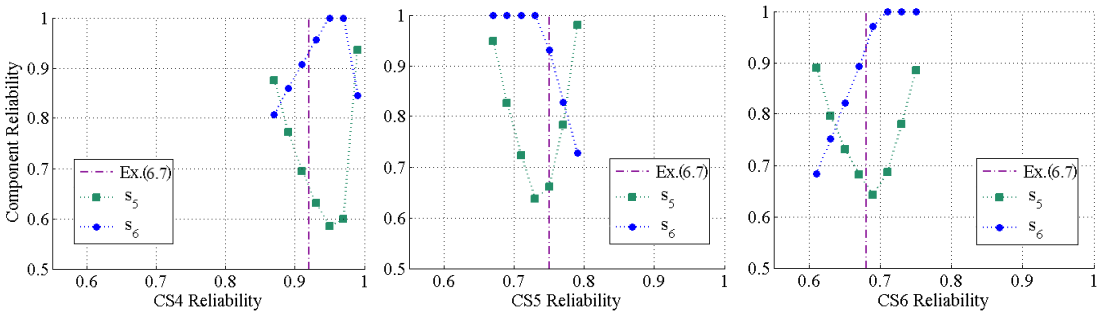
The equations can be simplified to the following.

$$\begin{aligned} &r_{s_5} \cdot r_{s_7} + r_{s_6} \cdot r_{s_7} - r_{s_5} \cdot r_{s_6} \cdot r_{s_7} = \alpha_4 \\ &0.5\, r_{s_5} \cdot r_{s_7} \cdot r_{s_8} + 0.5\, r_{s_6} \cdot r_{s_7} \cdot r_{s_8} = \alpha_5 \\ &0.3\, r_{s_5} \cdot r_{s_6} \cdot r_{s_8} + 0.3\, r_{s_6} \cdot r_{s_7} \cdot r_{s_8} + 0.4\, r_{s_5} \cdot r_{s_6} \cdot r_{s_7} \cdot r_{s_8} = \alpha_6 \end{aligned} \tag{6.22}$$

We can use the three constraints which have four unknowns as described in Subsection 6.3.3.

Example 6.7. A provider aims to determine the minimum reliability for the components s_5 and s_6 as a Sequence construct and has specified a threshold of 0.95 to select those components. The following values are aggregated from the reliability ratings of CS4, CS5 and CS6 respectively; {0.92, 0.75, 0.68}.

Using Matlab's function *fmincon* which by default uses the *active-set* algorithm,



(a) CS4 (b) CS5 (c) CS6

Figure 6.14: Changes in the Minimisation Solution of Components Reliability Determined from Reliabilities of More Complex Compositions

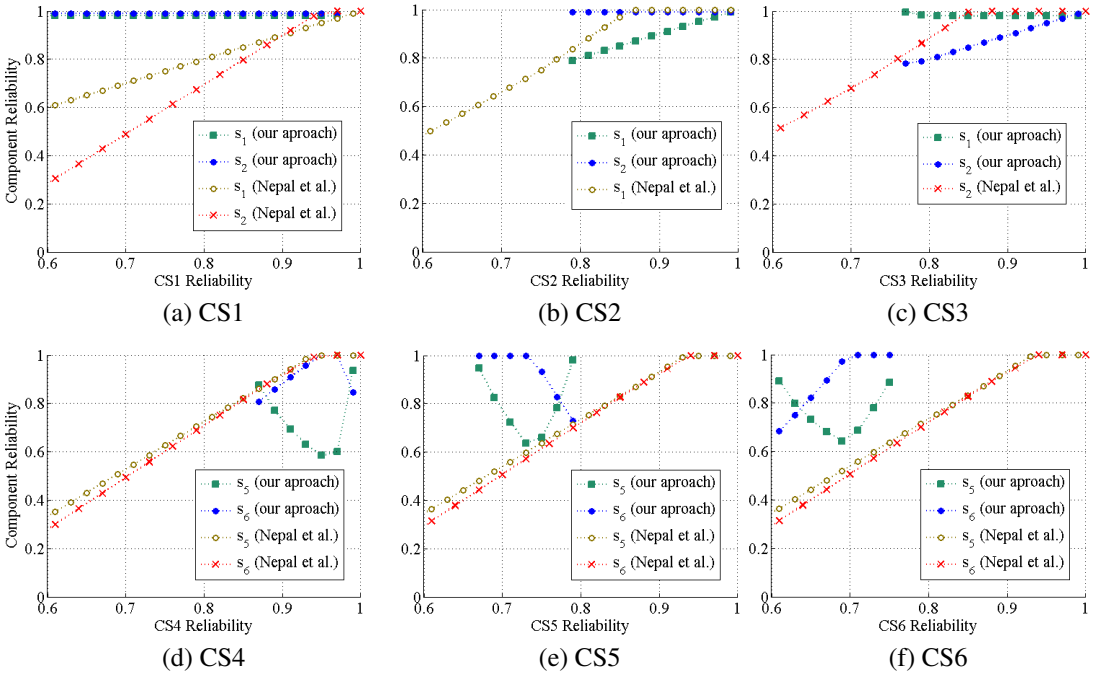
but also supports other alternative algorithms such as sequential quadratic programming or trust-region-reflective algorithm. The algorithm starting points for all variables are set at 0.5. The function gives the following solution {0.66, 0.93} for r_{s_5} and r_{s_6}. The solution indicates poor aggregated trustworthiness (0.61) which is well below the threshold.

Figure 6.14 shows the changes in the minimum values of component reliabilities when the reliability of each of CS4 to CS6 individually changes between values 0.55 and 1.0 while the remaining CSs are fixed at values stated in Example (6.7). The resulting changes of component reliabilities indicate that the blame for low trustworthiness may in some cases shift from one component to another as a result of a moderate change of a CS reliability. In other cases as in Example (6.6), component reliabilities don't change dramatically within the feasible solution space even with moderate changes in a composite service reliability. This indicates that the tolerance to CS reliability evaluation errors in the minimisation solution could vary from one case to another. The dash-dot line indicates the solution to Example (6.7) where all components' reliability values crossing the line are equal to the example solution.

Experimental Comparison with Existing Work

As discussed in Subsection 3.3.3 there is no existing work, to our knowledge, addressing collaborative determination of trustworthiness attributes of joint component services based on perceived or monitored attributes of their compositions. Existing work, in particular that from Nepal et al. [79], only considers determination of component attributes from an individual composite service. Figure 6.15 compares the results of the determination of the component reliabilities using simulations based on the approach we describe in this section to that from Nepal et al. (we will refer to it as Nepal's

approach). We implemented both approaches in MATLAB and used the algorithms described in this chapter and in [79] respectively. We compare the two approaches for several composite services (Figures 6.11 and 6.13) that vary in structure and size. The reliability value of the changing CS ranges between 0.6 and 1. The components for which the reliability values are calculated are those in Examples (6.6) and (6.7) i.e. s_1 and s_2 for CS1 to CS3 and s_5 and s_6 for CS4 to CS6.



(a) CS1 (b) CS2 (c) CS3

(d) CS4 (e) CS5 (f) CS6

Figure 6.15: Comparison of Our Collaborative Determination of Reliability to the Propagation Approach from Nepal et al. [79]

Although Nepal's approach includes exchange of trustworthiness data between entities in the service community, it does not support data correlation and the collective analysis of such data to derive attribute values. Therefore, their approach is used only to determine the attribute values of components of one composite service. Accordingly, sub-figures 6.15 (b and c) only show the result of one component in the changing composite service (i.e. CS2 and CS3 respectively). As in the figure, their propagated component reliability changes linearly as the reliability of the composition changes regardless of possible different causes of failure of a CS.

Nepal's approach also provides a result whether a CS reliability value is valid or not. As we discuss in this chapter, certain values of CS reliability can be infeasible based on the structure of the CS because the CS attribute value in that case would require component attribute values to be outside the attribute boundaries. This means that even if CS attribute values are erroneous because of bad reporting (e.g. CS attribute

values are within the boundaries but impossible to happen or CS attribute values are outside the allowed boundaries) Nepal's approach still distributes the values to components. This clearly should not be allowed as it would provide incorrect attribute values to components because the CS attribute values from which they would be calculated are incorrect in the first place. Note that in Nepal's approach, the values of the calculated component reliabilities that are outside the boundaries are reset to 0 if $r_i < 0$ or to 1 if $r_i > 1$, which can be seen in the sub-figures where the line representing the component reliability levels off once it reaches the value of 1. On the other hand, our approach detects infeasible CS attribute values and does not allow the calculation of attribute values for components when values for CS attributes are erroneous.

Additionally, Nepal's approach does not take into consideration the structure of the composition in the aggregation of reliability and other attributes. Consequently, the results from their approach are unaffected in sub-figures (d) to (f) despite the composite services are different. The results indicate our approach is more accurate because, in addition to being collaborative, it takes into consideration important variables in determining component reliability that are overlooked in the compared approach including the type of attribute, composition constructs and validity of attribute values.

6.5.2 Reputation

The calculation of reputation of components follows a similar approach to that of reliability but with consideration of the importance weights of components in each CS and the difference in the aggregation formulas for some constructs. This approach is based on our assumption as discussed in Chapter 2 that the reputations of the components of a CS are interdependent and that the reputation of a CS is influenced by the importance of each component as well as its reputation.

The reputation of components executed in a composite service with n components and not containing Exclusive Choice or Multi-Choice with Synchronised Merge constructs can be obtained using the linear Equation (6.7).

For a set of CS-based constraints we can represent the importance values of all components in a matrix. If the function $g(s_i, CS_j)$ is a function whose value is the weight of component s_i when it is a constituent of a composite service CS_j or zero otherwise,

$$g(s_i, CS_j) = \begin{cases} \omega_{i,j} & if\ s_i \in S_j \\ 0 & if\ s_i \notin S_j \end{cases} \tag{6.23}$$

where S_j is the set of components of a composite service CS_j, then the importance values can be represented using the following matrix where m is the number of composite services and z is the number of components.

$$\begin{pmatrix} g(s_1, CS_1) & g(s_2, CS_1) & \cdots & g(s_z, CS_1) \\ g(s_1, CS_2) & g(s_2, CS_2) & \cdots & g(s_z, CS_2) \\ \vdots & \vdots & \ddots & \vdots \\ g(s_1, CS_m) & g(s_2, CS_m) & \cdots & g(s_z, CS_m) \end{pmatrix} \tag{6.24}$$

Example 6.8. Suppose that according to the consumer reports, we have the following reputation values for CS1 to CS3 (Figure 6.11) respectively; {0.71, 0.81, 0.92}. The importance of each component ($\omega_{i,j}$) in each CS is displayed in the matrix below where a rows and columns represent individual composite services (CS1 to CS3) and components (s_1 to s_4) respectively.

$$\begin{pmatrix} 1.1 & 0.9 & 0.7 & 0 \\ 1.1 & 0 & 0 & 0.9 \\ 0 & 0.7 & 0 & 1.3 \end{pmatrix}$$

In order for components to be selected for a Sequence construct in the composite service their aggregated reputation is required to be above 0.92.

We have the system of constraints (based on Equation 6.7) and four variables. Accordingly, we can determine the minimum reputation of the set of components as described in Subsection 6.3.3. The resulting component reputations are {0.83, 0.89} which is below the threshold.

More complex constructs result in more complex constraints. The reputation in case of both Synchronized and Unsynchronized Parallel is aggregated using Equation (6.6) as their components are always executed. For composite services with Exclusive Choice constructs and Multi-Choice with Synchronised Merge constructs (as in Figure 6.13) the component reputations are represented using nonlinear constraints similar to the approach described for reliability.

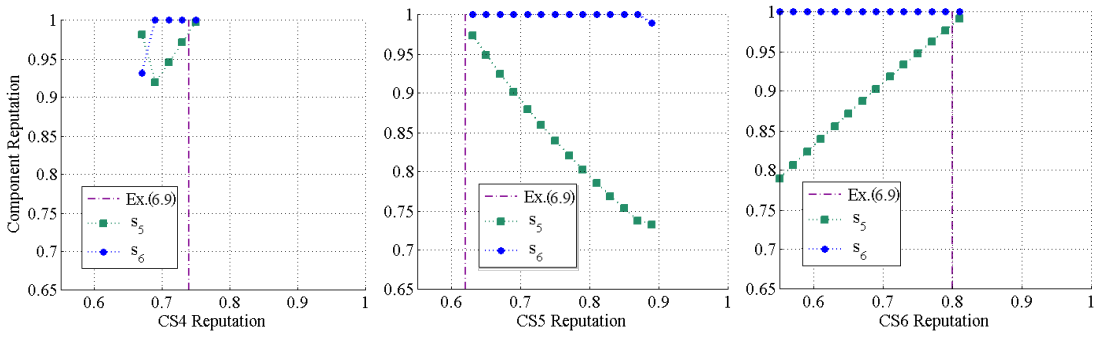
Therefore, the reputation of CS5 when s_5 and s_6 each has 50% chance of being executed in all CS5 executions can be represented using Equation (6.25) where ω_{s_5,cs_5} is the importance weight of the component s_5 in the composite service cs_5 and so on for the weights of the rest of the components in their CSs. As linear constraints are not available in this case, the problem is solved with nonlinear equality constraints.

$$p_{cs_5} = (p_{s_8})^{\omega_{s_8,cs_5}}(0.5\ (p_{s_5})^{\omega_{s_5,cs_5}} + 0.5\ (p_{s_6})^{\omega_{s_6,cs_5}})(p_{s_7})^{\omega_{s_7,cs_5}} \tag{6.25}$$

Example 6.9. A provider sets a threshold for the candidate components s_5 and s_6 at 0.9 for a Parallel construct. Based on consumer reports, the following reputation values were obtained for CS4 to CS6 respectively; {0.74, 0.62, 0.80}. The importance of each component ($\omega_{i,j}$) in each CS is according to the matrix below where each row and column represents a composite service (CS4 to CS6) and a component (s_5 to s_8) respectively and the element value is based in Equation (6.23).

$$\begin{pmatrix} 0.9 & 1.3 & 0.8 & 0 \\ 0.7 & 1.0 & 1.2 & 1.1 \\ 1.0 & 0.8 & 0.6 & 1.6 \end{pmatrix}$$

Using the system of nonlinear constraints representing the CSs and Matlab's *fmincon* function, the solution for the minimisation problem for the components s_5 and s_6 are as follows {0.98, 1.0} successively which is above the threshold.



(a) CS4 (b) CS5 (c) CS6

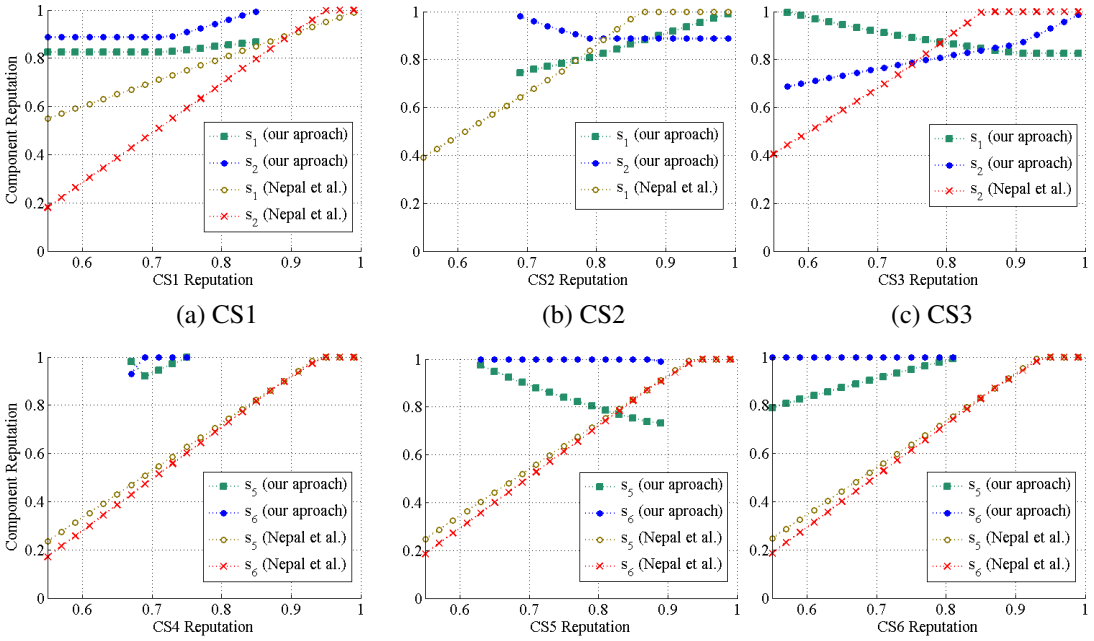
Figure 6.16: Changes in the Minimisation Solution of Components Reputation Determined from Reputations of Composite Services

Figure 6.16 shows the changes in the computed values of component reputations when the reputation of each of CS4 to CS6 individually changes between values 0.55 and 1.0 while remaining CSs don't change. The feasible results are when the component reputations are between 0 and 1.0. The dash-dot line indicates the solution to Example (6.9) where each component's line crosses the line of the example solution in all sub-figures at same point. The change in a CS reputation can indicate different trend of change for the reputation values of the components depending on the structure of the constraints i.e. the structure of the changing and other CSs.

Experimental Comparison with Existing Work

We compare our approach to Nepal's approach described in Subsection 3.3.3. Figure 6.17 compares results of determination of component reputations using simulations. We implemented both approaches in MATLAB and used the algorithms described in this chapter and in [79] respectively. We compare the two approaches for several composite services (Figures 6.11 and 6.13) that vary in structure and size. The components for which the reputation values are calculated are those in Examples (6.8) and (6.9) i.e. s_1 and s_2 for CS1 to CS3 and s_5 and s_6 for CS4 to CS6. The reputation value of the changing CS ranges between 0.6 and 1.

Nepal's approach determines the reputation of components from one composite service. Accordingly, sub-figures 6.17 (b and c) only show the result of one component in the changing composite service (i.e. CS2 and CS3 respectively). As in the figure, their propagated component reputation values change linearly as the reputation of a CS changes regardless of possible different causes of failure of the CS.



(a) CS1 (b) CS2 (c) CS3

(d) CS4 (e) CS5 (f) CS6

Figure 6.17: Comparison of Our Collaborative Determination of Reputation to the Propagation Approach from Nepal et al. [79]

Nepal's approach also provides a result whether the CS reputation value is valid or not. As we discuss in this chapter, certain values of a CS reputation can be infeasible based on the structure of the CS because the solution requires that some or all component attribute values be outside the attribute boundaries ($p_i < 0$ and $p_i > 1$). This

can occur for example due to bad reporting from collaborating CSs. In those cases the approach by Nepal et al. still distributes the values to components. This clearly should not be allowed as it would provide incorrect attribute values to components because the CS attribute values from which they would be calculated are incorrect. On the other hand, our approach detects infeasible CS attribute values and does not allow the calculation of attribute values for components when the values for the CS reputations are erroneous. In Nepal's approach, the values of the calculated component reputations that are outside the boundaries are reset to 0 if $p_i < 0$ or to 1 if $p_i > 1$, which can be seen in the sub-figures where lines representing component reputations levels off once it reaches the value of 1.

Furthermore, Nepal's approach does not take into consideration the structure of the composition in the aggregation of reputation. Consequently, the results from their approach are unaffected in sub-figures (d) to (f) despite the composite services are different.

The results indicate that our approach is more accurate because of its wider view of the status of the services due to being collaborative. In addition, it takes into consideration important variables in determining component reputation that are overlooked in the compared approach including the type of attribute, composition constructs and validity of attribute values. These variables are required to ensure accurate and fair propagation of reputation values to the components.

6.5.3 Response Time

The maximum expected response times of components can be determined from that of sequence CSs such as those in Figure 6.11 as in the example given below. A delay in a CS execution may be a result of a delay in one or more components. Component response times naturally vary depending on the characteristics of the service but the aim is to detect components that exceed advertised or agreed response times. While Sequence CSs can provide sufficient linear constraints, complex composite services provide non-linear constraints that are not usually sufficient to provide satisfactory solutions. For example, in Synchronised Parallel constructs, the components are executed in parallel but the next construct cannot commence its execution until all parallel components are complete. Therefore, the construct response time equals the longest of component response times.

Example 6.10. The response times of the simple composite services CS1 to CS3 in milliseconds are {200, 280, 270}. Using the approach described in Subsection 6.3.3 but as a maximisation problem for the response times of components

we find the maximum expected response times are 109 and 95 msec for s_1 and s_2 respectively as a Sequence construct.

6.6 Summary

The chapter describes a framework and detailed approaches to collaboratively determine the trustworthiness of joint distributed components of multiple composite services in service environments. The approach covers some of the important attributes of trustworthy services that are service reliability, reputation, response time and capacity. The trustworthiness of a composite service is based on consumers QoE reports regarding their satisfaction and on monitoring of reliability and response time.

In addition to characteristics of each attribute of a composite service, other aspects are also considered when computing the trustworthiness attributes of components. One such aspect is the structure of the CS process which determines the linearity and complexity of the constraint defined by the composite service. Another aspect is the degree of similarity and distribution of components between collaborating CSs which influences the certainty and preciseness of the predictions.

Multiple examples and experiments are used to examine and discuss the approach as well as illustrate its advantages and limitations. Experimental results of the approach are also compared to those from the related work. The results show that our approach provides better accuracy and support for factors that affect service trustworthiness including the type of attribute, composition constructs and validity of attribute values.

The approach can discover untrustworthy components and constructs with a varying range of tolerance to errors in composite service trustworthiness predictions. However, it is limited by the availability of other composite services using the candidate components and the exchange of knowledge about their structure and trustworthiness.

Chapter 7

Profitability and Cost Management of Trustworthy Composite Services

7.1 Introduction

Composition of business services is one of the main features of the service oriented computing, where existing services can be used to build higher level enterprise solutions. In many business cases, services are only competitive as part of an overall portfolio. In other cases, integrating new services increases profitability. Additionally, business innovations are frequently based on service integrations. Service composition has other advantages as well [156], including simplification of usage, enhancing reusability and improving modularisation as well as change management.

Profitability over short- and long-term is achieved through a number of measures including optimal pricing, cost efficiency and trustworthiness of provided services. Consumers do not necessarily buy the service with highest trustworthiness or lowest price. Therefore, attractiveness of a CS to consumers should be based on consumer market segments by providing multiple levels of trustworthiness and price targeting each segment. Cost efficiency improves profitability directly by creating a more flexible margin for profit and indirectly by affording to offer services at lower prices and hence more effectively compete as well as build consumer base.

Service composition techniques must be able to provide the most profitable, cost-efficient and trustworthy composite services for competitive service environments. In

Section 2.5 we described a scenario of multiple composite services offered by CSPs. CSPs aggregate component services and offer them to consumers as higher level business services. They have access to pools of component services of which many may provide the same functionality but with different levels of trustworthiness and cost. The roles of the CS components can also vary in their importance to its reputation. Additionally, CSPs may select a component service for multiple offered composite services. Therefore, CSPs require the management and balancing of component demand and admission taking into consideration market segments and priority of CSs in terms of their profitability. This chapter addresses enhancing profitability of those CSPs through novel approaches to the related issues of competitive CS pricing, cost efficiency and trustworthiness.

The chapter is structured as follows. Optimisation of pricing of composite services is described in Section 7.2. Section 7.3 introduces capacity dependent charging of component services and capacity determination based on factors that include those related to profitability such as consumer differentiation. Section 7.4 discusses simulation and experiments. A summary is discussed in Section 7.5.

7.2 Optimisation of Prices of Composite Services

Let us assume that m competing composite services are offered by competitors including one from our CSP. Each composite service j ($j = 1, .., m$) has a trustworthiness T and a price P. Consumers can choose between available compositions and make an overall evaluation of each CS based on a set of attributes by using *utility maximisation*. Attributes may differ in types and expected values between consumers depending on factors such as personal preferences. Bitran and Ferrer [126] classify utility into deterministic utility and stochastic utility; where deterministic utility is based on measurable choice such as in our case trustworthiness and price. Stochastic utility refers to independent factors such as incomplete information or errors in consumer perception. The deterministic utility is given by function $U_j(P, T, \alpha)$ where P is the price of the composite service j, T is its trustworthiness ($0 \leq T \leq 1$), α is the *price response function*. The price response function [157] determines how demand changes as a function of price and it indicates price sensitivity. The value α is unique for each service composition and market segment.

Price response function is based on assumptions about consumer behaviour. One of the models for consumer demand is called *willingness to pay*, which indicates the

maximum price a consumer is willing to pay for a service at a particular trustworthiness. A consumer who decides not to pay for a CS will choose one from a competitor or may not purchase any. Commonly a negative value (e.g. α=-0.07) is used to represent a price response function [126]; however, the function is typically not linear and tends to take the shape of a downward sloping curve. The slope of the curve at a given point may be affected by other factors in addition to the price change such as the distribution of consumers' willingness to pay, trustworthiness of the CS, and availability of alternative CSs from competitors. The two most common measures of price sensitivity [157] are:

- *Slope* of the price response function measures how demand changes due to a price change. It is equal to the change in demand divided by the change in price. The quality of this measurement reduces with larger changes in price as it assumes linearity of the price response function.
- *Elasticity* of the price response function is the ratio of percentage change in demand to the percentage change in price. Elasticity changes at different prices and tends to be highest around the market price.

The deterministic utility U_j for a composition j is calculated from its utility variables:

$$U_j = T + \alpha \cdot P \tag{7.1}$$

Multinomial Logit (MNL) is commonly used in economics as a consumer choice model. Using MNL in our case, the probability ρ of choosing the CSP's service instead of that of its competitors is as follows.

$$\rho = \frac{e^{U_1}}{\sum_{j=1}^{m} e^{U_j}} \tag{7.2}$$

where U_1 is the deterministic utility of the composite service from our CSP. Although ideally Equation (7.2) would include all competing providers, in practice it may be satisfactory to only consider a limited number of major competitors.

Finding the optimal price is an optimisation problem whose objective function is as follows.

$$\max\Big((P - C)\rho\Big) \tag{7.3}$$

where C is the cost of the composite service including the aggregated cost of its component services.

To create a new CS, the CSP may first select its market segments which determine prices and required trustworthiness values. Affordable trustworthiness and cost balance varies for each market segment. In that case, the CS can be offered at multiple price/trustworthiness values and the probability of purchase ρ_k for market segment k can be computed as follows.

$$\rho_k = \frac{e^{U_{k1}}}{\sum_{j=1}^{m} e^{U_{kj}}} \tag{7.4}$$

where $\sum_{j=1}^{m} e^{U_{kj}}$ is the sum of utilities for the CS and competing compositions for k.

7.3 Profitability and Cost in CS Provision

7.3.1 Capacity-Dependent Cost

We propose a dynamic component capacity management and capacity dependent component charging approach where:

- A component provider may adjust its component capacity according to dynamic capacity update requests from a CSP.
- A CSP pays for each component according to the capacity made available by the component provider (dynamic cost) in addition to its usage (fixed cost).

This approach can offer advantages compared to only per use based approach including: *(A)* the ability of CSPs to manage request admissions based on service availability, CS prioritisation and consumer differentiation. Consequently, it makes it easier to control pricing and profitability of composite services. *(B)* it can be more cost efficient to CSPs if discounting on capacity sizes is supported and more cost efficient to component providers since they can provide capacity adequate to the workload. *(C)* it helps maintain trustworthiness of CSs since the capacity required and offered is known to CS and component providers respectively. We calculate the cost c_i of a component service s_i as the sum of its fixed cost and dynamic cost; the fixed cost is based on usage u_i and the dynamic cost is based the provided capacity $\hat{y}_i$, as follows.

$$c_i = u_i \cdot c_u + \hat{y}_i \cdot c_y \tag{7.5}$$

where c_u is the fixed cost per usage and c_y is the cost of capacity at $\hat{y}_i$.

7.3.2 Priority-Based Capacity Determination and Admission

A CSP such as that in the scenario described in Section 2.5 may take several factors into consideration during the determination of capacity and admission of consumer requests. These factors include:

- The currently *ongoing and waiting executions* u_i and q_i of each component service s_i respectively according to the admitted requests.
- The existing *total capacity* $\hat{y}_i$ of each component service in terms of the number of possible concurrent executions as a component of one or more composite services. The capacity of component services in a composite service $\hat{y}_i$ are divided to two allocations; a shared capacity y_{ih} , and exclusive capacities assigned for each composite service y_{ij}. The goal of the allocations is to allow sharing of available resources while permitting priorities among CSs in admission when resources become scarce.
- The *maximum allowed execution time* $\hat{t}$ of the CS according to, for instance, a service level agreement, and *execution time* t_i of each component in order to predict the time of execution of all subsequent components in a CS workflow.
- The *constructs* in the CS workflow e.g. sequence, parallel, exclusive choice. A construct may contain two or more components running simultaneously. For example, in a parallel construct the minimum capacity of its components is taken as the capacity of the construct.
- The *choice of execution* of each component in the CS workflow. This may depend on the characteristics, interdependence or limited supply of some component services. For example, in an emergency composite service a fire service, an ambulance service or both may be required in certain executions. An example of limited capacity is where a highly trustworthy car rental service has limited supply. In that case more demand requires additional supply from other car rental service providers.
- The *priority of the requested CS* among offered composite services based for example on profitability. A prioritised composite service j is exclusively allocated a proportion y_{ij} of the capacity of each component.

– *Consumer differentiation* through ranking that can be based on market segments, loyalty, or other criteria. A minimum allocation of a component service's capacity can be assigned to a consumer rank. We consider 3 ranks *Gold (G)*, *Silver (V)* and *Bronze (B)*. The values y_{Gmin} and y_{Vmin} are fractions of CS exclusive capacities y_{ij} to maintain minimum capacities available exclusively to ranks G and V respectively. Note that $y_{ij} \geq (y_{Gmin} + y_{Vmin})$.

– *Component weighting*. Component services may vary in their importance to a CS as a whole. For example, in a travel service a user may not appreciate all components to the same extent such as car rental, health insurance, and flight booking. Therefore, component services differ in their contribution to the reputation of a CS. Each component s_i has a weight ω_i based on its importance. The weighting is useful when a component capacity is in full usage or close to becoming so. In that case a non-critical component can be excluded from a CS execution. This is particularly useful if the request would otherwise be rejected or when low remaining capacity can be saved for higher ranks. A threshold Ω_j is specified for the minimum weight of a component to be considered in the request admission decision. The set S_j is the set of components in the composite service j where $\omega_{ij} \geq \Omega_j$.

A request is either admitted immediately to execution, added to an execution queue, or possibly rejected depending on the above factors. Table 7.1 describes conditions for admission of a request immediately or to the queue.

Table 7.1: Conditions for Request Admission

Rank	**Admission**	**Conditions**
Gold	Immediate	$\forall s_i \in S_j : \{u_i < y_{ih}\ OR\ u_{ij} < y_{ij}\}$
	Queue	$\forall s_i \in S_j : \{u_i \geq y_{ih}\ AND\ u_{ij} \geq y_{ij}\ AND\ q_i < Q_j\}$
Silver	Immediate	$\forall s_i \in S_j : \{u_i < y_{ih}\ OR\ u_{ij} < (y_{ij} - y_{Gmin})\}$
	Queue	$\forall s_i \in S_j : \{u_i \geq y_{ih}\ AND\ u_{ij} \geq y_{ij}\ AND\ q_i < (Q_j - q_{Gmin})\}$
Bronze	Immediate	$\forall s_i \in S_j : \{u_i < y_{ih}\ OR\ u_{ij} < (y_{ij} - y_{Gmin} - y_{Vmin})\}$
	Queue	$\forall s_i \in S_j : \{u_i \geq y_{ih}\ AND\ u_{ij} \geq y_{ij} AND\ q_i < (Q_j - q_{Gmin} - q_{Vmin})\}$

As indicated in Table 7.1, G rank request for composite service j is admitted if one of the following conditions are met for every component in j:

- Current overall usage of a component service u_i is less than the overall shared capacity i.e. $u_i < y_{ih}$, or
- Availability in the capacity currently assigned exclusively to j i.e. $u_{ij} < y_{ij}$.

For ranks V and B, in addition to previous conditions for G, y_{ij} is reduced to allow minimum available capacity for higher ranks e.g. y_{Gmin}.

A request may be admitted to the queue if access was not given for immediate execution. In order to admit a request of rank G to the queue, the shared and the exclusive allocations of component services at the time of their execution (taking durations of executions of preceding components) must be in full use. Additionally, the queue for the component q_i should be less than the queue threshold for the composite service Q_j. The queue threshold depends on the maximum allowed time for the CS execution $\hat{t}$, its actual execution duration t_j, and the capacities of each construct in the composite service as follows.

$$Q_j = \frac{y_j(\hat{t} - t_j)}{\hat{t}} \tag{7.6}$$

where y_j is the average capacity of sequential constructs in the composite service. A construct may contain two or more components running simultaneously e.g. parallel executions, as described in Section 2.4. The minimum capacity of components running in parallel is taken as the capacity of the construct. Subsequently, the resulting sequential construct capacities are averaged. For ranks V and B the queue threshold is reduced by a proportion allocated to higher ranks e.g. q_{Gmin} and q_{Vmin} for rank B.

7.4 Simulation and Experiments

We simulated three composite services with outsourced component services based on those illustrated in Figure 2.2. The existence of components in multiple composite services results in a need to coordinate the usage of shared resources taking into consideration various factors described in Section 7.3.

The overall pattern of requests for each CS is based on datasets on requests for popular web pages [158]. Each pattern is applied to requests in a specific consumer rank for one CS. The rate of requests usually fluctuates between 10 to 50 requests per sec but the patterns particularly for the rank B include peaks at certain time durations reaching up to 150-250 requests per sec. Poisson distribution is used to simulate request arrivals. The execution duration for a CS depends on that of its components and on

how they are constructed. The execution time for each component is set at 100 msec on average. The value $\hat{t}$ for each CS is set at 25% more than average execution time t_j. The value is important in controlling the dynamic queue threshold as described in Subsection 7.3.2 and consequently affects the usage/capacity relationship.

Figures 7.1 and 7.2 show the results of the operation for a component service that exists in the three CSs. The capacity required is sent to the component provider which dynamically adjusts its capacity. In Figure 7.1 the resources of the component provider are flexible and the capacity can be adjusted for up to the maximum required. In case of Figure 7.2 the capacity is limited to a maximum of 300 concurrent executions. This results in the adjustment of the capacity of the CSs to that of its weakest link and consequently the rejection of excess requests of B and to a lesser extent of V rank. y_{Gmin} and y_{Vmin} are set at 30 and 15% of y_{ij}, and y_{ih} at 50% of $\hat{y}_i$.

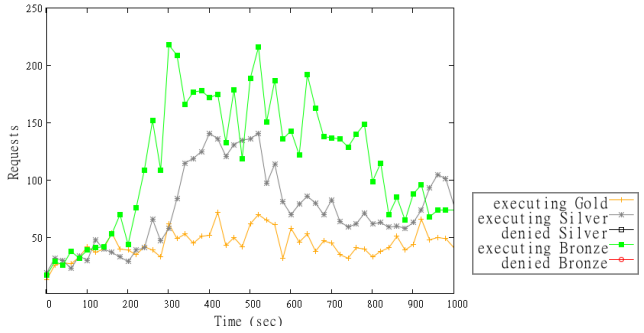


Figure 7.1: Usage in Flexible Capacity

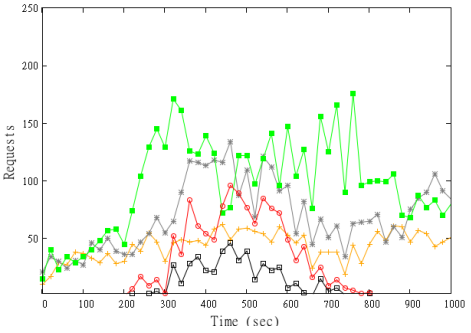


Figure 7.2: Usage in Limited Capacity

Figure 7.3 depicts denied requests for each of the CSs with exclusive capacities at 70, 20 and 10% of total nonshared component capacities for CS1, CS2, and CS3 respectively. Despite high exclusive capacity of CS1 there is still denied requests since a trade off is to be made between maximising usage of existing capacity v.s. denying some requests of higher priority CSs. Another option is the preemption or abandoning of ongoing low priority executions to admit higher priority requests in order to avoid denying prioritised requests.

Figure 7.4 shows the performance of the coordinated capacity determination and related resource management operations in msec taken for processing of requests. As in the figure there is a mild increase in the processing times as the rate of requests increases. However, the times do not seem to increase to a level that can cause a problem to the accuracy of the operations within moderate to high rate of requests.

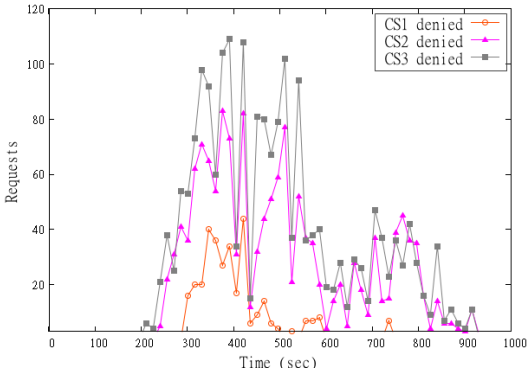


Figure 7.3: Denied Requests Per CS in Limited Capacity

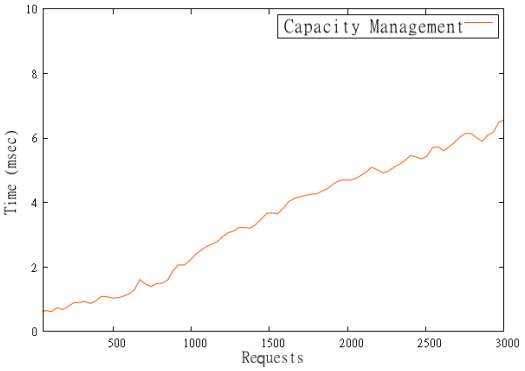


Figure 7.4: Performance of Capacity Management Operations

7.5 Summary

We presented novel approaches to managing short- and long-term profitability of composite services with the aim of maintaining as well as expanding consumer base in a competitive service environment where consumers purchase a composite service that maximises their utility. The management of cost and trustworthiness and the optimal pricing of a CS are made in view of compositions it will be competing with in the market.

This chapter described a technique for component service charging that is dependent on the capacity available to a composite service provider. The technique allows better management of costs and trustworthiness by the composite service provider and consequently a better control of its profitability.

The admission of requests can be prioritised based on the capacity available for the request's rank and the priority of the requested CS. The aim is to help enhance profitability of CSPs especially in cases of limited availability due to high request rates or constrained capacities. Additionally, the communication of capacity requirements to component service providers helps maintain the trustworthiness and cost efficiency of composite services.

Chapter 8

Conclusion

8.1 Summary of Findings

This thesis describes techniques for establishing and maintaining trustworthy composite services where sets of component services each with a distinct functionality are offered by multiple component providers. Alternative services are available for selection for each task in a composition. The alternative component services, which can support a structured business process, may have different current values and patterns of change of their trustworthiness attributes.

The thesis describes aggregation techniques for trustworthiness attributes based on the structure and other characteristics of composite services. The techniques consider the variation in importance of component services, the probability of executions and business process models which are not considered in the related work. The techniques also protect from unaddressed threats such as the presence of an untrustworthy component in a service composition. Additionally, it describes the aggregation of trustworthiness attributes for a composite or component service into a common trustworthiness value. An approach to unified view of service trustworthiness during service selection is described which weighs all attributes together in the selection decision.

A trustworthiness software module with a novel architecture is developed for the implementation of monitoring and prediction techniques and algorithms. The module includes a rules engine for the generation of service ratings and a trust engine that calculates and updates trustworthiness levels based on the service ratings. A fast algorithm described in the thesis allows trustworthiness to be updated continuously based on new

ratings/metrics received and on the recency of past ratings. A custom genetic algorithm is proposed to determine the optimal set of component services based on their trustworthiness and cost.

One of the important contributions of this work is the application of the concept of attack surfaces to service oriented computing. In particular, it describes the composition of attack surfaces of atomic services together with attack surfaces of their business processes. The research takes a multifaceted view of the concept of attack surfaces and provides a classification of security aspects of services based on the protection methods and the exploitability of the service resources. An existing methodology called the structured assurance case model is used to measure the security attributes of operations and data items belonging to entry and exit points of components and their re-evaluation as components are incorporated into service compositions.

A framework and a detailed approach for the determination of component trustworthiness are developed based on the trustworthiness attributes of collaborating composite services. The approach can identify untrustworthy components and detect the trustworthiness of candidate components to be selected for new composite services. The composite services jointly invoke distributed component services. In order to determine a trustworthiness attribute for a set of component services, we first set up a system of equations. Each equation corresponds to the structure of the process of a particular composite service. The variables in those equations correspond to attributes of the components. The approach can discover untrustworthy components with a range of tolerance to errors in composite service trustworthiness evaluations. However, it is limited by the availability of joint components in other composite services and the exchange of knowledge about their structure and trustworthiness.

This thesis addresses enhancing profitability of composite service providers who offer services to markets that contain competing services. The approach takes into consideration the related issues of competitive CS pricing, cost efficiency and trustworthiness. The techniques assume that consumers' willingness to pay for a CS is based on main factors that include service price, its trustworthiness, and availability of alternative services from competitors.

Service providers including CSPs aim to maintain the trustworthiness of their services during execution particularly with non-deterministic request arrivals and peak times and the possible uncertainty of component service execution in some composite service workflows. Meanwhile, resource allocation efficiency is also needed to reduce costs and improve profitability. We provide an approach to make a balance between

these two objectives. We describe techniques for priority-based capacity determination and admission including CSs with shared resources, CS priorities, consumer differentiation and capacity adjustment. A service request may be admitted immediately for execution, added to an execution queue, or possibly rejected depending on certain stated considerations. We consider this work as a first step in advancing the topic of trust-oriented resource management and further research is required.

8.2 Future Work

A useful feature for the trustworthiness module to work on in the future is to extend its rating repository to include evidence data to ensure that it takes into consideration factors such as the strength and accuracy of evidence of metrics received. The evidence evaluation can be based on the structured assurance case model and its applications. Enhancement of the module with such features may create opportunities for its commercial exploitation. The module functions may be offered as a service to evaluate trustworthiness of organisations who offer online services. Additionally, the correlation of QoE and consumer perception with QoS and other objective attributes in order to predict consumer perception based on actual level of trustworthiness is a related area that can help automate and enrich trustworthiness evaluations that rely on results of consumer surveys.

We foresee that the collaborative component trustworthiness approach described in the thesis can also be extended to service trustworthiness attributes other than those discussed. This would depend on the techniques used to aggregate each attribute from components to composite services. This requires further investigation. Additionally, evaluation of the approach in a real-world service environment involving providers and consumers would help identify improvements and further features as well as analysing its business usage scenarios.

The problem of how to ensure reliable evaluation of attack surface based security metrics of integrated services that are independent and distributed is an area of future research. The aim is to protect composite services from discrepancies (possibly intentional) between independent attack surface measurements and the actual security positions of component services.

In the future we aim to extend the work on profitability of trustworthy services to allow the integration of the described cost efficiency and trustworthiness techniques in composite services with security access control policies and mechanisms.

Another interesting related area of future research is the extension of business process modelling languages such as BPMN to support the representation, transformation and enforcement of trustworthiness, reputation, privacy and security requirements for component and composite services.

Trustworthiness-based ranking and filtering of vulnerabilities and security threats to services is a practical topic of future research. The research could aim to evaluate the trustworthiness of threat submissions and improve the quality of data in threat repositories. The inclusion of the threat updates in the trustworthiness prediction and its aggregation with trustworthiness attributes is also a topic of future research.

Glossary

Access Right. A description of the type of authorized interactions a subject can have with a resource e.g. read or write.

Atomic Service. An indivisible software service that encapsulates few coherent business or technical functionalities.

Attack Surface. The set of entry points and exit points and their characteristics in a system which can be used to determine the relevant potentially exploitable system resources.

Availability. The degree to which a system or component is operational and accessible when required for use.

Business Process. A set of activities or tasks performed by a business that transforms data, materials or business commitments, and produces a service or product.

Business Process Execution Engine. A software service that provides a runtime execution environment to support the execution of business process instances according to the process definitions.

Camouflaging. In the context of attacks against trust and reputation systems, low trustworthiness of a component is hidden through insertion into a trustworthy composite service.

Component Service. A software service meant to interact with other components, encapsulating certain functionality or a set of functionalities. A component service has a service interface and conforms to a prescribed behaviour common to the service architecture [159].

Composite Service (CS). An integration of lower level component services such as atomic services, assembled together in order to automate a business process.

Composite Service Provider (CSP). A service provider who aggregates component services possibly from other providers and offers them to consumers as higher level business services.

Confidentiality. The property of a system that certain information can only be read by authorised entities.

Context. The circumstances that form the setting for an event or a statement and in terms of which it can be fully understood and assessed [160].

Encapsulation. A software development technique that consists of isolating a system function or a set of data and operations on those data within a module and providing precise specifications for the module [161].

Distributed Hash Table (DHT). In a distributed system, hash function is used to map keys such as file names into points in a logical coordinate space. The coordinate space is partitioned dynamically among the peers in the system such that every peer covers a region in that space. Peers are responsible for storing (key, value) pairs the keys of which are hashed into a point located within their region [51].

Free-riding. In the context of attacks against trust and reputation systems, a component service unjustifiably benefits from the high trustworthiness ratings of a composite service.

Integrity. The degree to which a system or component prevents unauthorized access to or modification of an application or data [161].

Metric. A quantitative measure of the degree to which a system, component, or process possesses a given attribute [161].

Non-repudiation. A method by which the sender of data is provided with proof of delivery and the recipient is assured of the sender's identity, so that neither can later deny having processed the data [162].

Reliability. The ability of a system or component or service to perform its required functions under stated conditions for a specified period of time [161].

Response time. The time perceived by a requester of a service between the time of the service invocation and the time at which the response to the invocation is received.

Reputation. The information available about an entity based on consumer satisfaction ratings that can be used together with other types of information in determining the entity's trustworthiness.

Self-Promoting. An attack against a trust and reputation system in which the attacker manipulates the reputation of an entity by unjustifiably increasing it [56].

Service. A unit of solution logic to which service-orientation has been applied to a meaningful extent. It is the application of service-orientation design principles that distinguish a unit of logic as a service compared to units of logic that may exist only as objects or components [163].

Service Construct. A subset of a composite service process that consists of tasks that follow common invocation rules such as order and triggering events.

Service Level Agreement (SLA). A formal agreed binding contract between a service consumer and a service provider specifying characteristics such as quality of service.

Service-Oriented Computing (SOC). A computing paradigm that utilizes services as fundamental elements for developing distributed applications [63].

Slandering. Attackers manipulate the reputation of others by providing false reports to decrease the reputation of the victims [56].

Subprocess. A set of activities that have a logical order that meet certain purpose. A Subprocess is a process whose functionality is part of a larger process.

Sybil Attack. An attack on reputation systems by forging multiple identities in peer-to-peer networks to gain greater influence on the system [57].

Trust. A relationship between two or more entities that indicates context-based expectations from an entity towards another in relation to reliance in accomplishing a certain action at a certain quality or outcome.

Trustworthiness. Trustworthiness of an entity is the level of trust that the trusting entity or its agent has in that entity.

Trustworthiness Attribute. A measurable attribute of a service whose variation in value can affect the objective and/or perceived trustworthiness of the service. A value for an attribute can be measured by aggregating subjective consumer ratings or objective metrics e.g. using automated monitoring systems.

trustworthiness attribute rating. A value generated by the trustworthiness module's rules engine to normalise metrics and consumer ratings. The trust engine uses the trustworthiness attribute ratings related to a service to compute its trustworthiness value.

Whitewashing. Attackers exploit a reputation system vulnerability to improve their reputation and once they restore their reputation, they continue their malicious behaviour [56].

Workflow. A process or a movement of information from one activity to another.

Vulnerability. A flaw or weakness in a system's design, implementation, or operation that can be exploited to violate the system's security [17].

Bibliography

[1] H. Elshaafi and D. Botvich, "Aggregation of Trustworthiness Properties of BPMN-based Composite Services," *Proc. 17th IEEE Int. Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (CAMAD)*, pp. 383-387, Sep 2012.

[2] H. Elshaafi, J. McGibney and D. Botvich, "Trustworthiness Monitoring and Prediction of Composite Services," *Proc. 17th IEEE Symposium on Computers and Communication (ISCC)*, Jul 2012.

[3] H. Elshaafi, J. McGibney and D. Botvich, "Attack Surface Based Security Metric Framework for Service Selection and Composition," *submitted to Inderscience Journal of Autonomous and Adaptive Systems*, 2014.

[4] H. Elshaafi and D. Botvich, "Trustworthiness Inference of Multi-tenant Component Services in Service Compositions," *FTRA Journal of Convergence (JoC)*, 4(1):31-37, Mar 2013.

[5] H. Elshaafi and D. Botvich, "Optimisation Based Collaborative Determination of Component Trustworthiness in Service Compositions," *J. Security and Communication Networks*, John Wiley & Sons, 2014.

[6] H. Elshaafi, J. McGibney and D. Botvich, "Profitability and Cost Management of Trustworthy Composite Services," *Proc. 9th Int. Conf. Trust, Privacy & Security in Digital Business (TrustBus)*, pp. 179-191, Sep 2012.

[7] Z. Malik and A. Bouguettaya, "RATEWeb: Reputation Assessment for Trust Establishment among Web services," *VLDB J.*, 18(4):885-911, Feb 2009.

[8] A. Singhal, T. Winograd and K. Scarfone, "Guide to Secure Web Services," National Institute of Standards and Technology (NIST), Tech. Rep., 2007.

[9] H. Takabi, J. Joshi and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *IEEE Security & Privacy*, Mar 2010.

[10] W. Hasselbring and R. Reussner, "Toward Trustworthy Software Systems," *IEEE Computer*, 39(4):91-92, Apr 2006 .

[11] A. Jøsang, E. Gray and M. Kinateder, "Simplification and Analysis of Transitive Trust Networks," *Web Intelligence and Agent Systems*, 4(2):139-161, 2006.

[12] E. Chang, T. Dillon and F. Hussain, *Trust and Reputation for Service-Oriented Environments: Technologies For Building Business Intelligence And Consumer Confidence*, 2005.

[13] D. Artz and Y. Gil, "A Survey of Trust in Computer Science and the Semantic Web," *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58-71, Jun 2007.

[14] D. Gambetta, "Can We Trust Trust?", *Trust: Making and Breaking Cooperative Relations*, pp. 213-238, Basil Blackwell Ltd, 1988.

[15] T. Grandison and M. Sloman, "A survey of Trust in Internet Applications," *IEEE Communications Surveys & Tutorials*, 3(4):2-16, 2000.

[16] A Jøsang, R. Ismail and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," *Decision Support Systems J.*, Elsevier, 43(2):618-644, Mar 2007.

[17] R. Shirey, "Internet security glossary," version 2, Internet Engineering Task Force (IETF), RFC 4949, Aug 2007.

[18] M. Blaze, J. Feigenbaum and J. Lacy, "Decentralized Trust Management," *Proc. 17th IEEE Symp. Security and Privacy*, 1996.

[19] Business Process Model and Notation (BPMN) 2.0, Object Management Group, http://www.omg.org/spec/BPMN/2.0.2.

[20] M. Weske, *Business Process Management. Concepts, Languages, Architectures*, 2nd ed., Springer, 2012.

[21] N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View," BPM Center Report BPM-06-22, http://BPMcenter.org, 2006.

[22] Workflow Patterns Initiative, http://www.workflowpatterns.com.

[23] G.-G. Lee and H.-F. Lin, "Customer perceptions of e-service quality in online shopping," *Int. J. Retail & Distribution Management*, 33(2):161-176, 2005.

[24] H. Li and R. Suomi, "A Proposed Scale for Measuring E-service Quality," *Int. J. u-and e-Service*, 2(1):1-10, 2009.

[25] G. J. Udo, K. K. Bagchi and P. J. Kirs, "An assessment of customers' e-service quality perception, satisfaction and intention," *Int. J. Information Management*, 30(6):481-492, 2010.

[26] C. Sohn and S. K. Tadisina, "Development of e-service quality measure for internet-based financial institutions," *Total Quality Management & Business Excellence*, 19(9):903-918, 2008.

[27] Activiti BPM Platform, http://www.activiti.org.

[28] K. Saeedi, L. Zhao and P. R. F. Sampaio, "Extending BPMN for Supporting Customer-Facing Service Quality Requirements," *Proc. 8th IEEE Int. Conf. Web Services*, pp. 616-623, 2010.

[29] C. J. Pavlovski and J. Zou, "Non-Functional Requirements in Business Process Modeling," *Proc. 5th Asia-Pacific Conf. Conceptual Modelling*, pp. 103-112, 2008.

[30] O. Moser, F. Rosenberg and S. Dustdar, "Non-Intrusive Monitoring and Service Adaptation for WS-BPEL," *Proc. 17th Int. Conf. World Wide Web*, 2008.

[31] T. Kelly, *Arguing Safety - A Systematic Approach to Managing Safety Cases*, PhD thesis, York University, Department of Computer Science, Sep 1998.

[32] T. Rhodes, F. Boland, E. Fong and M. Kass, "Software Assurance Using Structured Case Models," *Journal of Research of the National Institute of Standards and Technology*, 115(3):209-216, May 2010.

[33] H. Elshaafi, J. McGibney and D. Botvich, "Trustworthiness Monitoring of Dynamic Service Compositions," *Proc. 6th Workshop on Enhanced Web Service Technologies*, ACM, pp. 25-29, Sep 2011.

[34] H. Elshaafi, J. McGibney and D. Botvich, "Business Driven Optimisation of Service Compositions," *Proc. 7th Int. Conf. Next Generation Web Services Practices*, IEEE, 2011.

[35] H. Elshaafi and D. Botvich, "Trustworthiness Inference of Multi-tenant Component Services in Service Compositions," *Proc. 4th FTRA Int. Conf. Computer Science and its Applications (CSA)*, Springer, pp. 301-312, Nov 2012.

[36] Trusted Computing Group (TCG), http://www.trustedcomputinggroup.org.

[37] J. Sabater, *Trust and reputation for agent societies*, PhD thesis, Universitat Autònoma de Barcelona, 2002.

[38] T. D. Huynh, *Trust and Reputation in Open Multi-Agent Systems*, PhD dissertation, University of Sourthampton, 2006.

[39] L. Xiong, L. Liu and I. C. Society, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Trans. Knowledge and Data Engineering*, 16(7):843–857, Jul 2004.

[40] A. Das and M. M. Islam, "SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multiagent Systems," *IEEE Trans. Dependable and Secure Computing*, 9(2):261-274, 2012.

[41] J. C. Ely and J. Välimäki, "Bad reputation," *Quarterly J. Economics*, 118(3):785-814, 2003.

[42] H. Yu, M. Kaminsky, P. B. Gibbons and A. D. Flaxman, "SybilGuard: defending against sybil attacks via social networks," *IEEE/ACM Transactions on Networking*, 16(3):576-589, 2008.

[43] S. Xu, X. Li, T. P. Parker and X. Wang, "Exploiting trust-based social networks for distributed protection of sensitive data," *IEEE Transactions on Information Forensics and Security*, 6(1):39-52, 2011.

[44] D. J. Kim, D. L. Ferrin and H. R. Rao, "A trust-based consumer decision-making model in electronic commerce: The role of trust, perceived risk, and their antecedents," *Decision Support Systems J.*, Elsevier, 44(2):544-564, Jan 2008.

[45] A. Erciş, S. Ünal, F. B. Candan and H. Yıldırım, "The Effect of Brand Satisfaction, Trust and Brand Commitment on Loyalty and Repurchase Intentions," *Procedia - Social and Behavioral Sciences*, 58(12):1395-1404, Oct 2012.

[46] J. McGibney and D. Botvich, "Establishing Trust Between Mail Servers to Improve Spam Filtering," *Autonomic and Trusted Computing*, 2007.

[47] WS-Trust specification document, OASIS, http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html, Feb 2009.

[48] A. Carroll, M. Juarez, J. Polk and T. Leininger, "Microsoft Palladium: A Business Overview," Microsoft white paper, June, 2002.

[49] C. Mundie, P. de Vries, P. Haynes and M. Corwine, "Trustworthy Computing," Microsoft white paper, 2002.

[50] G. Zacharia and P. Maes, "Trust Management through Reputation Mechanisms," *Applied Artificial Intelligence*, 14(9):881-907, 2000.

[51] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks," *Proc. 12th Int. conf. World Wide Web*, 2003.

[52] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *Proc. IEEE Trans. Parallel and Distributed Systems*, 18(4):460-473, 2007.

[53] W. Teacy, J. Patel, N. Jennings and M. Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," *J. Autonomous Agents and MultiAgent Systems*, 12(2):183-198, 2006.

[54] Z. Noorian and M. Ulieru, "The State of the Art in Trust and Reputation Systems A Framework for Comparison," *J. Theoretical and Applied Electronic Commerce Research*, 2010.

[55] O. Khalid, S. U. Khan, S. A. Madani, K. Hayat and M. I. Khan, "Comparative study of trust and reputation systems for wireless sensor networks," *J. Security and Communication Networks*, John Wiley & Sons, 6(6):669-688, 2013.

[56] K. Hoffman, D. Zage and C. Nita-Rotaru, "A survey of Attack and Defense Techniques for Reputation Systems," *ACM Computing Surveys*, 42(1):1-31, Dec 2009.

[57] J. R. Douceur, "The Sybil attack," *Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 251-260, Springer, Mar 2002.

[58] A. Jøsang and J. Golbeck, "Challenges for Robust Trust and Reputation Systems," *Proc. 5th Int. Workshop on Security and Trust Management*, Sep 2009.

[59] F. G. Mármol and G. M. Pérez, "Security Threats Scenarios in Trust and Reputation Models for Distributed Systems," *Computers & Security*, 28(7):545-556, Oct 2009.

[60] K. Walsh and E. Sirer, "Fighting Peer-to-Peer SPAM and Decoys with Object Reputation," *Proc. 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, 2005.

[61] Z. Malik and A. Bouguettaya, *Trust Management for Service Oriented Environments*, Springer, 2009.

[62] Y. Yang, Y. Sun, S. Kay and Q. Yang, "Defending Online Reputation Systems against Collaborative Unfair Raters through Signal Modeling and Trust," *Proc. 24th ACM Symp. Applied Computing*, 2009.

[63] M. P. Papazoglou, P. Traverso, S. Dustdar and F. Leymann, "Service-Oriented Computing: a Research Roadmap," *Int. J. Cooperative Information Systems*, 17(2):223, 2008.

[64] M. P. Singh, "Trustworthy Service Composition: Challenges and Research Questions," *Proc. 1st Int. Conf. Autonomous Agents and Mutli-agent Systems*, 2002.

[65] F. Skopic, *Dynamic Trust in Mixed Service-oriented Systems*, PhD thesis, Technischen Universität Wien, 2010.

[66] F. Skopik, D. Schall and S. Dustdar, "Modeling and Mining of Dynamic Trust in Complex Service-Oriented Systems," *Information Systems J.*, 35(7):735-757, Nov 2010.

[67] N. Dragoni, "Toward Trustworthy Web Services - Approaches, Weaknesses and Trust-By-Contract Framework," *2009 IEEE/WIC/ACM Int. Joint Conf. Web Intelligence and Intelligent Agent Technology*, 2009.

[68] I. Pranata, R. Athauda and G. Skinner, "Modeling Decentralized Reputation-Based Trust for Initial Transactions in Digital Environments," *ACM Trans. Internet Technol.*, 12(3), 2013.

[69] G. Spanoudakis and S. LoPresti, "Web Service Trust: Towards a Dynamic Assessment Framework," *Proc. 4th Int. Conf. Availability, Reliability and Security*, 2009.

[70] S. Majithia, A. Shaikh, O. F. Rana and D. W. Walker, "Reputation-based Semantic Service Discovery," *Proc. 13th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Jul 2004.

[71] E. M. Maximilien and M. P. Singh, "Agent-based Trust Model Involving Multiple Qualities," *Proc. 4th Int. Joint Conf. Autonomous Agents and Multiagent systems*, 2005.

[72] S. Paradesi, P. Doshi and S. Swaika, "Integrating Behavioral Trust in Web Service Compositions," *2009 IEEE Int. Conf. Web Services*, pp. 453-460, 2009.

[73] C. Hang and M. P. Singh, "Trustworthy Service Selection and Composition," *ACM Trans. Autonomous and Adaptive Systems*, 6(1):1-18, 2011.

[74] L. Li and Y Wang, "Trust Evaluation in Composite Services Selection and Discovery," *2009 IEEE Int. Conf. Services Computing*, 2009.

[75] M. Mehdi, N. Bouguila and J. Bentahar, "Trustworthy Web Service Selection Using Probabilistic Models," *2012 IEEE 19th Int. Conf. Web Services (ICWS)*, pp. 24-29, Jun 2012.

[76] T. Ciszkowski, W. Mazurczyk, Z. Kotulski, T. Hoßfeld, M. Fiedler and D. Collange, "Towards Quality of Experience-based reputation models for future web service provisioning," *Telecommunication Systems*, 51(4):283-295, Dec 2012.

[77] A. E. Arenas, B. Aziz and G. C. Silaghi, "Reputation Management in Collaborative Computing Systems," *Security and Communication Networks*, John Wiley & Sons, 3(6):546-564, 2010.

[78] J. Abawajy, "Determining Service Trustworthiness in Intercloud Computing Environments," *Proc. 10th Int. Symp. Pervasive Systems, Algorithms and Networks*, 2009.

[79] S. Nepal, Z. Malik and A. Bouguettaya, "Reputation Propagation in Composite Services," *Proc. 2009 IEEE Int. Conf. Web Services* pp. 295-302, Jul 2009.

[80] P. Meland, "Service Injection: A Threat to Self-Managed Complex Systems," *Proc. 9th IEEE Int. Conf. Dependable, Autonomic and Secure Computing*, pp. 1-6, 2011.

[81] S. Wen, Q. Li, L. Yue, A. Liu and C. Tang, "Reputation Distribution Based on Structure-Related Importance in Services Composition," *Proc. IEEE 8th Int. Conf. e-Business Engineering*, Oct 2011.

[82] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," *Proc. 2010 ACM/IEEE 32nd Int. Conf. Software Engineering*, pp. 35-44, May 2010.

[83] E. Bertino, L. Martino, F. Paci and A. Squicciarini, *Security for Web Services and Service-Oriented Architectures*, Springer, 2010.

[84] A. D. Brucker and I. Hang, "SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes," *Proc. ACM symposium on access control models and technologies*, 2012.

[85] A. Rodríguez, E. Fernández-Medina and M. Piattini, "A BPMN Extension for the Modeling of Security Requirements in Business Processes," *IEICE - Trans. Information and Systems*, 90-D(4):745-752, 2007.

[86] M. Menzel and C. Meinel, "SecureSOA: Modelling Security Requirements for Service-oriented Architectures," *Proc. 2010 IEEE Int. Conf. Services Computing*, 2010.

[87] Aniketos (Secure and Trustworthy Composite Services) Project, http://www.aniketos.eu.

[88] OWASP Code Review Guide V11, The Open Web Application Security Project (OWASP), https://www.owasp.org, 2008.

[89] H. Shahriar and M. Zulkernine, "Taxonomy and Classification of Automatic Monitoring of Program Security Vulnerability Exploitations," J. Systems and Software, 84(2):250-269, Feb 2011.

[90] M. Blaze, S. Kannan, I. Lee, O. Sokolsky, J. M. Smith, A. D. Keromytis and W. Lee, "Dynamic Trust Management," *IEEE Computer*, 42(2):44-52, Feb 2009.

[91] K. Khan and J. Han, "A Security Characterisation Framework for Trustworthy Component Based Software Systems," *Proc. 27th Int. Conf. Computer Software and Applications*, 2003.

[92] B. Schneier, "Attack Trees", *Dr. Dobb's Journal*, 24(12):21-29, Dec 1999.

[93] M. Dacier, Y. Deswarte and M. Kaâniche, "Models and tools for quantitative assessment of operational security," *Proc. 12th IFIP Information systems security Conf.*, pp. 177-186, 1996.

[94] O. Sheyner, J. Haines, S. Jha, R. Lippmann and J. M. Wing, "Automated generation and analysis of attack graphs," *Proc. 2002 IEEE Symposium on Security and Privacy*, pp. 273-284, 2002.

[95] S. Bistarelli, F. Fioravanti and P. Peretti, "Defense trees for economic evaluation of security investments," Proc. 1st Int. Conf. Availability, Reliability and Security (ARES), pp. 416-423, Apr 2006.

[96] K. S. Edge, "A Framework For Analyzing And Mitigating The Vulnerabilities Of Complex Systems Via Attack And Protection Trees", PhD thesis, Air Force Institute of Technology, 2007.

[97] A. Vorobiev and J. Han, "Security Attack Ontology for Web Services," *Proc. 2nd IEEE Int. Conf. Semantics, Knowledge and Grid*, Nov 2006.

[98] P. Mell, K. Scarfone and S. Romanosky, "CVSS: A Complete Guide to the Common Vulnerability Scoring System Version 20," Forum of Incident Response and Security Teams (FIRST), Jun 2007.

[99] W. Jansen, "Directions in Security Metrics Research," National Institute of Standards and Technology, Tech. Rep., 2009.

[100] J. Bau and J. C. Mitchell, "Security Modeling and Analysis," *IEEE Security and Privacy*, 9(3):18-25, Jun 2011.

[101] M. Howard, J. Pincus and J. Wing, "Measuring relative attack surfaces," *Proc. Workshop on Advanced Developments in Software and Systems Security*, 2003.

[102] P. K. Manadhata and J. M. Wing, "An Attack Surface Metric," *IEEE Transactions on Software Engineering*, 37(3):371-386, May 2011.

[103] T. Boland and E. Fong, "Toward a Preliminary Framework for Assessing the Trustworthiness of Software," National Institute of Standards and Technology, Tech. Rep., 2010.

[104] C. B. Weinstock, H. F. Lipson and J. Goodenough, "Arguing Security - Creating Security Assurance Cases," https://buildsecurityin.us-cert.gov/articles/knowledge/assurance-cases, Carnegie Mellon University, 2007.

[105] M. Ouedraogo, D. Khadraoui, H. Mouratidis and E. Dubois, "Appraisal and reporting of security assurance at operational systems level," *Journal of Systems and Software*, 85(1):193-208, Jan 2012.

[106] F. Swiderski and W. Snyder, *Threat modeling*, Microsoft Press, Jul 2004.

[107] A. Shostack, "Experiences Threat Modeling at Microsoft," *Proc. 1st Int. Modeling Security Workshop*, Toulouse, 2008.

[108] A. Shostack, *Threat Modeling: Designing for Security*, Wiley, Apr 2014.

[109] A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt*, Addison Wesley, 2007.

[110] J. Cardoso, "Quality of Service for Workflows and Web Service Processes," *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308, Apr 2004.

[111] M. Jaeger, G. Rojec-Goldmann and G. Muhl, "QoS Aggregation for Web Service Composition Using Workflow Patterns," *Proc. 8th IEEE Int. Enterprise Distributed Object Computing Conf.*, 2004.

[112] S. Hwang, H. Wang, J. Tang and J. Srivastava, "A Probabilistic Approach to Modeling and Estimating the QoS of Web-services-based Workflows," *Information Sciences*, 177(23):5484-5503, Dec 2007.

[113] S.-Y. Hwang, E.-P. Lim, C.-H. Lee and C.-H. Chen, "Dynamic Web Service Selection for Reliable Web Service Composition," *IEEE Trans. Services Computing*, 1(2):104-116, Apr 2008.

[114] V. Grassi and S. Patella, "Reliability prediction for service-oriented computing environments," *Internet Computing*, 10(3):43-49, 2006.

[115] A. Parasuraman, V. A. Zeithaml and L. L. Berry, "SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality," *J. Retailing*, 64(1):12-40, 1988.

[116] Y. Wang and J. Vassileva, "A Review on Trust and Reputation for Web Service Selection," *Proc. 27th Int. Conf. Distributed Computing Systems*, 2007.

[117] K. C. Lee, J. H. Jeon, W. S. Lee, S. Jeong and S. Park, "QoS for Web Services: Requirements and Possible Approaches," W3C Working Group Note, http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/, 2003.

[118] N. Limam and R. Boutaba, "Assessing Software Service Quality and Trustworthiness at Selection Time," *IEEE Trans. Software Engineering*, 36(4):559-574, 2010.

[119] A. Yüksel and F. Yüksel, "The Expectancy-Disconfirmation Paradigm: A Critique," *J. Hospitality & Tourism Research*, 25(2):107-131, May 2001.

[120] M. Jaeger and G. Mühl, "QoS-based Selection of Services: The Implementation of a Genetic Algorithm," *KiVS (Kommunikation in Verteilten Systemen) Workshop*, 2007.

[121] A. Huang, C.-W. Lan and S. Yang, "An optimal QoS-based Web service selection scheme," Inf. Sci. J., 179(19):3309-3322, Sep 2009.

[122] Z. U. Rehman, F. K. Hussain and O. K. Hussain, "Towards Multi-criteria Cloud Service Selection," *Proc. 5th Int. Conf. Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 44-48, Jun 2011.

[123] F. Lécué, "Optimizing QoS-Aware Semantic Web Service Composition," *Proc. 8th Int. Semantic Web Conf.*, 2009.

[124] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Trans. Software Engineering*, 33(6):369-384, Jun 2007.

[125] I. Ng, "The Pricing and Revenue Management of Services," *J. Revenue and Pricing Management*, 7(2):231-232, Jun 2008.

[126] G. Bitran and J. Ferrer, "On Pricing and Composition of Bundles," *Production and Operations Management*, 16(1):93-108, 2007.

[127] J. Chung and V. Rao, "A General Choice Model for Bundles with Multiple-Category Products: Application to Market Segmentation and Optimal Pricing for Bundles," *J. Marketing Research*, 40(2):115-130, 2003.

[128] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas and A. Delis, "Flexible use of cloud resources through profit maximization and price discrimination," *Proc. IEEE 27th Int. Conf. Data Engineering (ICDE)*, pp.75-86, Apr 2011.

[129] Marbukh, V.; Mills, K., "Demand Pricing & Resource Allocation in Market-Based Compute Grids: A Model and Initial Results," Proc. 7th Int. Conf. Networking (ICN), pp. 752-757, Apr 2008.

[130] G. Gallego and R. Phillips, "Revenue Management of Flexible Products," *Manufacturing & Service Operations Management*, 6(4):321-337, 2004.

[131] A. Petrick, J. Gönsch, C. Steinhardt and R. Klein, "Dynamic Control Mechanisms for Revenue Management with Flexible Products," *Computers & Operations Research*, 37(11):2027-2039, Nov 2010.

[132] B. Wu, C. Chi, Z. Chen, M. Gu and J. Sun, "Workflow-based Resource Allocation to Optimize Overall Performance of Composite Services," *Future Generation Computer Systems*, 25(3):199-212, Mar 2009.

[133] D. F. García, J. García, J. Entrialgo, M. García, P. Valledor, R. García and A. M. Campos, "A QoS Control Mechanism to Provide Service Differentiation and Overload Protection to Internet Scalable Servers," *IEEE Trans. Services Computing*, 2(1):3-16, Jan 2009.

[134] B. Urgaonkar, *Dynamic Resource Management in Internet Hosting Platforms*, PhD thesis, Computer Science Dept., University of Massachusetts Amherst, Sep 2005.

[135] V. Bianco, L. Lavazza, S. Morasca and D. Taibi, "A Survey on Open Source Software Trustworthiness," *IEEE Software*, 2011.

[136] C. Steel, R. Nagappan and R. Lai, *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services and Identity Management*, Pearson Education Inc., 2006.

[137] OSGi Alliance Specifications, http://www.osgi.org/Specifications.

[138] Drools - The Business Logic integration Platform, http://www.jboss.org/drools.

[139] P. D. Marinescu and G. Candea, "Efficient Testing of Recovery Code Using Fault Injection," *ACM Transactions on Computer Systems*, 29(4), 2011.

[140] C. Fu, B. G. Ryder, A. Milanova and D. Wonnacott, "Testing of Java Web Services for Robustness," *Proc. 2004 ACM SIGSOFT Int. Symposium on Software testing and analysis*, pp. 23-34, 2004.

[141] S. H. Kuk and H. S. Kim, "Robustness testing framework for Web services composition," *Proc. 2009 IEEE Asia-Pacific Services Computing Conf. (APSCC)*, pp. 319-324, 2009.

[142] G. Friedrich, M. Fugini, E. Mussi, B. Pernici and G. Tagni, "Exception Handling for Repair in Service-Based Processes," *IEEE Transactions on Software Engineering*, 36(2):198-215, Mar 2010.

[143] A. Avižienis, J.-C. Laprie, B. Randell and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable Sec. Comput.*, 1(1):11-33, 2004.

[144] A. Chuvakin and G. Peterson, "Logging in the Age of Web Services," *IEEE Security and Privacy*, 7(3):82-85, Jun 2009.

[145] Argumentation Metamodel (ARM), OMG (Object Management Group) Systems Assurance Taskforce, Version 10-Beta 1, OMG Document No. ptc/2010-08-36, http://www.omg.org/spec/ARM, Aug 2010.

[146] E. Luke, "Using Claims, Arguments and Evidence: A Pragmatic View - and tool support in ASCE," http://www.adelard.com, 2008.

[147] OWASP Application Security Verification Standard (ASVS), Open Web Application Security Project (OWASP), 2013.

[148] E. Triantaphyllou, B. Shu, S. N. Sanchez and T. Ray, "Multi-Criteria Decision Making: An Operations Research Approach," *Encyclopedia of Electrical and Electronics Engineering*, 15:175-186, John Wiley & Sons, 1998.

[149] J. Gennari and D. Garlan, "Measuring Attack Surface in Software Architecture," Carnegie Mellon University, Tech. Rep., 2012.

[150] K. Tsipenyuk and G. McGraw, "Seven Pernicious Kingdoms : A Taxonomy of Software Security Errors," *IEEE Security & Privacy*, 3(6):81-84, 2005.

[151] Common Weakness Enumeration (CWE), http://cwe.mitre.org.

[152] The Ten Most Critical Web Application Security Vulnerabilities, OWASP, https://www.owasp.org/index.php/OWASP_Top_10, 2013.

[153] Core Security Inc., http://www.coresecurity.com.

[154] Applicure Ltd, http://www.applicure.com.

[155] A. Doupé, M. Cova, and G. Vigna, "Why Johnny can't pentest: An analysis of black-box web vulnerability scanners," *Proc. 7th Int. Conf. Detection of intrusions and malware and vulnerability assessment*, pp. 111-131, 2010.

[156] A. Arsanjani, "Toward a Pattern Language for Service-Oriented Architecture and Integration," IBM DeveloperWorks, 2005.

[157] R. Philips, *Pricing and Revenue Optimization*, Stanford Business Books, 2005.

[158] Trending Topics, http://www.trendingtopics.org.

[159] W3C Web Services Glossary, http://www.w3.org/TR/ws-gloss, World Wide Web Consortium (W3C), 2004.

[160] Oxford Dictionaries, http://www.oxforddictionaries.com.

[161] IEEE Standard Glossary of Software Engineering Terminology, IEEE, 1990.

[162] National Information Systems Security (INFOSEC) Glossary, NSTISSI No. 4009, National Security Agency, US, 1999.

[163] SOA Glossary, http://serviceorientation.com/index.php/soaglossary/index, Arcitura Education Inc., 2007.