DISSERTATION

Dynamic Trust in Mixed Service-oriented Systems

Models, Algorithms, and Applications

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften

unter der Leitung von

Prof. Dr. Schahram Dustdar
Distributed Systems Group
Institut für Informationssysteme (E184)
Technische Universität Wien

und

Prof. Dr. Frank Leymann
Institut für Architektur von Anwendungssystemen (IAAS)
Universität Stuttgart

eingereicht an der

Technischen Universität Wien
Fakultät für Informatik

von

Florian Skopik
Matr.Nr. 0325234
Franz Schubertstraße 43
3701 Großweikersdorf

Wien, Mai 2010

Abstract

The way people interact in collaborative and social environments on the Web has evolved in a rapid pace over the last few years. Services have become a key-enabling technology to support collaboration and interactions. Pervasiveness, context-awareness, and adaptiveness are some of the concepts that emerged recently in service-oriented systems. A system is not designed, deployed, and executed; but rather evolves and adapts over time. This paradigm shift from closed systems to open, loosely coupled Web services-based systems requires new approaches to support interactions.

A large amount of social networks and collaboration platforms are realized as service-oriented systems enabling flexible compositions of services and support of interactions. Usually, interactions between entities in such systems do not only span supporting software services, but also human actors, participating in the network. A mixed service-oriented system is therefore composed of human and software services. In open eco-systems where massive collaborations between thousands of humans and services take place, flexibly joining and leaving the environment, their interactions are highly dynamic and often influenced by the role and reputation of collaboration partners. Therefore, introducing the concept of system-managed trust in such environments helps to keep track of the dynamics and to improve the overall benefit of social systems and people's collaboration performance. In this context, trust is not defined from a common security perspective, but from a social and behavioral point of view. The trustworthiness of actors is not approved by certificates and negotiations, but relies on the personalized outcome of previous interactions in terms of reliability, dependability and success.

This research aims at covering, models, algorithms and applications that support the management of trust in such mixed service-oriented systems environments. In particular, this thesis deals with (i) *trust and reputation models in service-oriented architectures*, relying on behavior patterns and dealing with temporal aspects; (ii) *trust mining and prediction on the Web*, including community mining, member clustering and ranking; and (iii) *trust-based adaptations in service-centric applications*, such as context-aware discovery and ranking of experts, selective information disclosure in collaborations, and trust-aware group formations in large-scale networks.

Kurzfassung

Internet-basierte kollaborative und soziale Plattformen haben in den letzten Jahren grundlegend das Verhalten und die Interaktionsmuster von Benutzern beeinflusst. Services wurden zur Schlüsseltechnologie um Kollaborationen und Interaktionen zu unterstützen. Mobilität von Benutzern, Kontextabhängigkeit und Adaptivität von Systemen sind nur einige der Konzepte, welche kürzlich in service-orientierten Architekturen Einzug gehalten haben. Kontextabhängige Systeme werden nicht in sequentiellen Schritten entwickelt, konfiguriert, und betrieben, sondern dynamisch über die Zeit an neue Anforderungen angepasst. Dieser Paradigmenwechsel von geschlossenen Systemen zu offenen, lose gekoppelten Web Service-basierten Systemen benötigt neue Ansätze um Interaktionen zu unterstützen.

Ein Großteil heutiger sozialer und kollaborativer Plattformen wird als Service-orientierte Systeme realisiert und unterstützt die flexible Komposition von Services und deren Interaktionen. In der Regel finden in solchen Systemen Interaktionen nicht nur zwischen Software Services statt, sondern auch zwischen Personen, welche am Netzwerk teilhaben. Ein "Mixed System" ist daher eine Komposition aus Menschen und Software-basierten Services. In offenen Systemen, wo Kollaborationen zwischen mehreren tausend Menschen und Services stattfinden, welche dieser Umgebung flexibel beitreten und diese auch wieder verlassen, sind Interaktionen von Dynamik geprägt und auch meist von Rollen und Reputation von Kollaborationspartnern beeinflusst. Die vorgelegte Dissertation führt das Konzept des System-verwalteten Vertrauens ein, welches dabei helfen soll, den Überblick und Kontrolle über die Dynamik in solchen Systemen zu behalten. In dieser Arbeit wird Vertrauen von einem sozialen Standpunkt betrachtet; im Gegensatz zu einer üblichen Security-Perspektive. Vertrauen in einen Akteur wird dabei nicht durch Zertifikate und Verhandlungen hergestellt, sondern beruht maßgeblich auf den Ergebnissen früherer Interaktionen und deren individuellen Nutzen, im Sinne der Zuverlässigkeit, Glaubwürdigkeit und des Erfolgs von Kollaborationen.

Diese Arbeit beschäftigt sich mit Modellen, Algorithmen und Applikationen, welche die Verwaltung von Vertrauen in "mixed" Service-orienterten Systemen ermöglichen. Insbesondere behandelt diese Dissertation (i) *Vetrauens- und Reputationsmodelle in Service-orienterten Architekturen*, welche auf Verhaltensmuster aufbauen und auch temporale Aspekte berücksichtigen; (ii) *Vertrauensbildung und Vorhersage im Web*, einschließlich Gruppenbildung und dem Erstellen von Ranglisten in Internet Communities; und (iii) *Vertrauensbasierte Adaption in Service-orientierten Applikationen*, wie zum Beispiel kontextabhängige Suche und Bewertung von Experten, selektives Offenlegung von Informationen in Kollaborationen, und vertrauensabhängige Gruppenbildung in Netzwerken großen Umfangs.

Acknowledgements

First and foremost, I would like to thank my adviser Prof. Schahram Dustdar for his continuous support and the chance to do a PhD thesis in the Distributed Systems Group. He created an environment that allowed me to develop my own ideas, and reliably kept me on track to turn them into reality. Additionally, I would also like to thank Prof. Frank Leymann from the University of Stuttgart for valuable comments to improve this thesis and for being my examiner.

I thank my colleague Daniel Schall who introduced me to the concepts of Mixed Systems and provided invaluable support during the last years. He taught me how to ask the right questions and express my ideas. He showed me different ways to approach a research problem and the need to be persistent to accomplish any goal. I would also like to acknowledge all my colleagues in our group, especially the members of the 320er team, Christoph Dorn, Lukasz Juszczyk, Harald Psaier, and Martin Treiber. It was a great time working together with you on various papers, software prototypes, and – last but not least – project proposals.

I am grateful for the constant financial support from the EU project COIN (FP7-216256), funding the research discussed in this dissertation.

Finally, I owe special gratitude to my family for continuous and unconditional support: To my parents for all the opportunities they made available to me, and for the support they have given me along the way; and to Tina for her enduring patience, understanding, and love. This dissertation is dedicated to you.

Florian Skopik
May, 2010
Vienna, Austria

For my parents and Tina

Publications

Parts of the work presented in this dissertation have been published in the following conference papers, journal articles, and technical reports.

1. Skopik F., Schall D., Dustdar S. (2010). *Modeling and Mining of Dynamic Trust in Complex Service-oriented Systems*. Information Systems. Elsevier. *forthcoming*.

2. Skopik F., Schall D., Dustdar S. (2010). *Supporting Network Formation through Mining under Privacy Constraints*. 10th Annual International Symposium on Applications and the Internet (SAINT), July 19-23, 2010, Seoul, South Korea. IEEE.

3. Psaier H., Skopik F., Schall D., Dustdar S. (2010). *Behavior Monitoring in Self-healing SOA*. 34th IEEE Computer Software and Applications Conference (COMPSAC), July 19-23, 2010, Seoul, South Korea. IEEE.

4. Skopik F., Schall D., Dustdar S., Sesana M. (2010). *Context-Aware Interaction Models in Cross-Organizational Processes*. 5th International Conference on Internet and Web Applications and Services (ICIW), May 9-15, 2010, Barcelona, Spain. IEEE.

5. Skopik F., Schall D., Dustdar S. (2010). *Innovative Human Interaction Services - Final Specification*. FP7-216256 COIN Technical Report D4.5.1b, May 2010.

6. Treiber M., Skopik F., Schall D., Dustdar S., Haslinger S. (2010). *Context-aware Campaigns in Social Networks*. 5th International Conference on Internet and Web Applications and Services (ICIW), May 9-15, 2010, Barcelona, Spain. IEEE.

7. Skopik F., Schall D., Dustdar S. (2010). *Adaptive Information Disclosure in a Dynamic Web of Trust*. Technical Report TUV-184-2010-03, April 2010, *under review for publication*.

8. Schall D., Skopik F., Dustdar S. (2010). *Trust-based Discovery and Interactions in Expert Networks*. Technical Report TUV-1841-2010-01, April 2010, *under review for publication*.

9. Skopik F., Schall D., Dustdar S. (2010). *Trust-based Adaptation in Complex Service-oriented Systems*. 15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), March 22-26, 2010, University of Oxford, UK. IEEE.

10. Skopik F., Schall D., Dustdar S. (2010). *Trustworthy Interaction Balancing in Mixed Service-oriented Systems*. 25th ACM Symposium On Applied Computing (SAC), March 22-26, 2010, Sierre, Switzerland. ACM.

11. Skopik F., Schall D., Dustdar S. (2010). *Trusted Interaction Patterns in Large-scale Enterprise Service Networks*. 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP), February 17-19, 2010, Pisa, Italy. IEEE.

12. Skopik F., Schall D., Dustdar S. (2009). *Start Trusting Strangers? Bootstrapping and Prediction of Trust*. 10th International Conference on Web Information Systems Engineering (WISE), October 05-07, 2009, Poznan, Poland. Springer.

13. Skopik F., Schall D., Dustdar S. (2009). *The Cycle of Trust in Mixed Service-oriented Systems*. 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), August 27-29, 2009, Patras, Greece. IEEE.

14. Skopik F., Truong H.-L., Dustdar S. (2009). *Trust and Reputation Mining in Professional Virtual Communities*. 9th International Conference on Web Engineering (ICWE), June 24-26, 2009, San Sebastian, Spain. Springer.

15. Skopik F., Truong H.-L., Dustdar S. (2009). *VieTE - Enabling Trust Emergence in Service-oriented Collaborative Environments*. 5th International Conference on Web Information Systems and Technologies (WEBIST), March 23-26, 2009, Lisbon, Portugal. INSTICC.

16. Skopik F., Schall D., Truong H.-L., Dustdar S. (2009). *Innovative Human Interaction Services Specification*. FP7-216256 COIN Technical Report D4.5.1a, January 2009.

17. Skopik F., Truong H.-L., Dustdar S. (2008). *Current and Future Technologies for Collaborative Working Environments*. Study for the European Space Agency, ESA ITT Number AO/3-12280/07/NL/CB, May 2008.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

B4P	BPEL for People (*also:* BPEL4People)
BPEL	*see WS-BPEL*
BPMN	Business Process Modeling Notation
COIN	Collaboration and Interoperability of Networked Enterprises (EU project)
ECA	Event-Condition-Action
FOAF	Friend of a Friend
HPS	Human Provided Services
HTML	Hyper Text Markup Language
IM	Instant Messaging
IP	Internet Protocol
OEM	Original Equipment Manufacturer
PVC	Professional Virtual Community
QoS	Quality of Service
REST	Representational State Transfer
RFS	Request for Support
SaaS	Software as a Service
SBS	Software-based Service
SDI	Selective Dissemination of Information
SLA	Service Level Agreement
SMS	Short Message Service
SOA	Service-oriented Architecture
SOAP	*originally:* Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
VieTE	Vienna Trust Emergence Framework
VO	Virtual Organization
VoIP	Voice over IP
WS	Web Service
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Service Description Language
WSMX	Web Services Modelling Execution Environment
XFN	XHTML Friends Network
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XPDL	XML Process Definition Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation

CHAPTER 1

Introduction

Trust is a social good to be protected just as much as the air we breathe or the water we drink. When it is damaged, the community as a whole suffers; and when it is destroyed, societies falter and collapse.

Sissela Bok, "Lying: Moral Choice in Public and Private Life"
New York, Patheon Books, 1978.

Contents

1.1 Motivation

The support of people's interactions on collaborative and social Web platforms has evolved in a rapid pace over the last few years. Services have become a fundamental pillar of modern Web architectures. Today's support of pervasiveness, context-awareness, and adaptiveness leads to a paradigm shift from traditional closed environments to open, loosely coupled, service-oriented systems. However, as these systems become increasingly complex, they also require new approaches to support discovery of members and their interactions.

While others mainly focus on either communities of humans (that may be supported by communication services), or compositions of pure software components, we deal with a mixture of humans and Web Services in one harmonized environment. A *Mixed Service-oriented System* comprises human- and software services that can be flexibly and dynamically composed to perform various kinds of activities. Therefore, interactions in such a system do not only span humans, but also software services. The Human-Provided Services (HPS) framework [106] enables human interactions through standardized SOA technologies, such as WSDL, SOAP, and various WS-* protocols. Thus, humans and Web services can be seamlessly unified in one environment using the same grounding technologies. Such Mixed Systems may be applied in a wide range of domains. Imagine a support community, where people can ask questions and other participants provide answers. In today's Internet forums there are plenty of questions that have been either already answered before or can be easily answered by referencing manuals and FAQs[1]; and thus could be automatically processed by Web services without human intervention. Recent efforts already try to integrate intelligence in the Web through sophisticated reasoning technologies (see Wolfram Alpha[2]). The advantages of introducing services in human networks are obvious: both humans and services can be composed to complement each other in beneficial ways. For instance, while services are able to process and repeat simple procedures very fast, humans are able to process more demanding requests.

However, with today's technologies it is hard – if not impossible – to keep track of the dynamics in such open loosely coupled large-scale systems. Therefore, introducing the concept of *trust* that dynamically emerges and periodically adapts, deems to be an intuitive concept to address this challenge [7, 58]. Considering trust relations when selecting people for communication or collaboration, services to be utilized, and resources to be applied leads to more efficient cooperation and compositions of human- and software services [111]. In contrast to many others, we do not discuss trust from a security perspective. In this work we follow another view that is related to how much humans or other systems can rely on services to accomplish their tasks [102].

This notion of *dynamic social trust* is closely coupled to common concepts from the social network domain, including behavior modeling, interaction patterns, and incentive engineering. In this thesis, we focus on enabling and supporting these concepts, and the creation and application of trust and reputation models in *Mixed Service-oriented Systems*. However, this is no pure theoretical work. We also present the technical grounding using existing SOA technologies and WS-Standards, and evaluate their applicability through simulations and real world data analysis.

1.2 Research Challenges and Questions

In order to establish and manage trust in mentioned Mixed Systems, we deal with the following challenges in this work:

- *Meaning and Definition of Trust in Mixed Systems.*
  - What are representative Mixed Systems and why is trust management required?
  - Which ones of the various definitions of trust are applicable in Mixed Systems environments?
  - What are the mechanisms to manage the context of trust?
- *Determination of Trust in Service-oriented Networks.*
  - What are the advantages of calculating trust based on objectively collected data?
  - What data needs to be collected to determine trust and how is this realized in today's SOAs?

[1] frequently asked questions

[2] Wolfram Alpha: `http://www.wolframalpha.com`

  - What mechanisms are available to infer trust based on observed and collected data?

- *Enabling Cold Start and Bootstrapping of Trust Management Systems.*

  - How can we guarantee the usability of the system to attract a critical mass of users, also immediately after start-up, where no or not much data has been collected?

  - How can newcomers be flexibly and quickly integrated in an existing Web of trust?

- *Investigating Advances through Trust-based Adaptations in Service-centric Applications.*

  - How can the management of trust relations enhance group formation processes, dynamic collaboration partner discovery, or the selective disclosure of information in terms of privacy?

  - How can trust management capabilities be integrated into existing service-centric applications?

1.3 Dependencies

Research is never conducted alone and independent from others. In order to address the mentioned challenges, we rely on some preceding work. In particular, we make extensive use of the following prerequisites:

- *SOA Infrastructures.* This includes the usage of fundamental SOA techniques, such as WSDL and SOAP (and supporting frameworks); but also more advanced standards, including WS-Addressing. We do not discuss these fundamental technologies, but assume the reader is familiar with them. For instance, see [95] for comprehensive discussions and explanations.

- *Interaction Logging and Processing in SOA.* Interaction Monitoring in SOA, i.e., intercepting SOAP calls, their temporal correlation, and calculation of higher level interaction metrics has been well addressed in [104]. We use these results and base our work on top of such captured interaction networks and their data sets.

- *Human-Provided Services (HPS).* The HPS framework is the means to enable convenient human interaction logging using appropriate SOAP interceptors and logging services. We discuss how HPSs support the emergence of trust, however, we do not address fundamental HPS design decisions. This information can be found in the HPS literature, for instance [106].

1.4 Contributions

Major contributions (with respect to discussed research challenges) and their interrelations are depicted in Figure 1.1. In detail they comprise the following novelties:

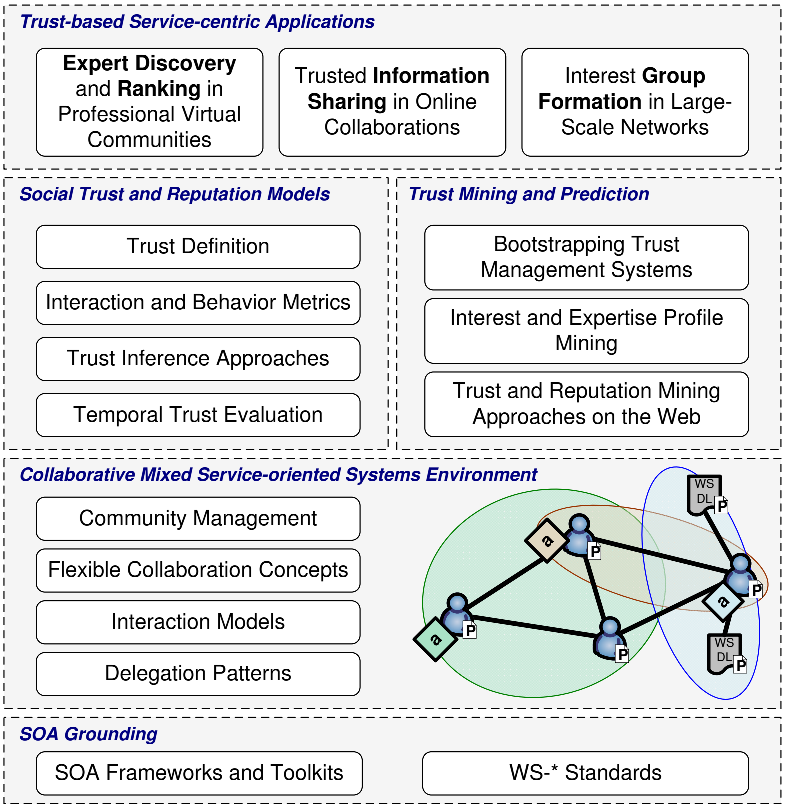

Trust-based Service-centric Applications
Expert Discovery and Ranking in Professional Virtual Communities
Trusted Information Sharing in Online Collaborations
Interest Group Formation in Large-Scale Networks
Social Trust and Reputation Models
Trust Definition
Interaction and Behavior Metrics
Trust Inference Approaches
Temporal Trust Evaluation
Trust Mining and Prediction
Bootstrapping Trust Management Systems
Interest and Expertise Profile Mining
Trust and Reputation Mining Approaches on the Web
Collaborative Mixed Service-oriented Systems Environment
Community Management
Flexible Collaboration Concepts
Interaction Models
Delegation Patterns
WS DL
P
a
SOA Grounding
SOA Frameworks and Toolkits
WS-* Standards

Figure 1.1: The big picture of combined contributions.

- *Collaborative Mixed Service-oriented Systems Environment.* Mixed Systems have been extensively discussed in [104]. However, we revisit flexible collaboration concepts with respect to motivating research challenges, and highlight the means of interactions and delegation patterns for trust determination.

- *Social Trust and Reputation Models.* Our models, presented in Part I of this thesis, rely on previous interaction behavior in terms of reliability, dependability and success. We establish context-awareness of trust relations, and introduce concepts for the automatic establishment and periodic updates by applying a rule based interpretative approach.

- *Trust Mining and Prediction Models.* These models, discussed in Part II, are applied if no interactions have been observed in the past. They deal with the cold start problem and with bootstrapping trust management systems. We present trust and reputation mining algorithms, as well as concepts for automatic interest profile determination and similarity measurements to predict future trust relations.

- *Trust-based Service-centric Applications.* Trust-based applications are in the focus of Part III. Flexible expert discovery mechanisms and ranking models, selective information disclosure in formation scenarios, and group formations accounting for privacy in the Web of Social Trust are among demonstrated examples. We discuss the integration of trust and reputation concepts in service-oriented frameworks.

- *SOA Grounding.* In contrast to many others, for instance in the agent domain, we also care for the technical grounding and the realization of our concepts using state of the art SOA technologies, such as SOAP, WSDL and various WS-* standards. This contribution spans all chapters within this thesis.

1.5 Organization of the Thesis

This dissertation includes the accumulated contributions from original research papers that have been published during PhD studies. The single contributions are extended, unified, and discussed in context of a big picture. Corresponding papers are referenced in the following structural description of the thesis.

After motivating our work in Chapter 1, this dissertation continuous with Chapter 2, an extensive analysis of the state of the art regarding trust and reputation in service-oriented environments. The main body is divided into three parts according to contributions depicted in Figure 1.1:

- Part I: Trust and Reputation Models in SOA
  - *Chapter 3: The Cycle of Trust in Mixed Service-oriented Systems* [111]: This work starts with establishing a detailed understanding of the notion of trust in mixed service-oriented systems, including a first technical grounding in SOA. We discuss a fundamental approach to dynamically determine trust from the system's point of view, and further outline a user perspective.
  - *Chapter 4: Trustworthy Interaction Balancing in Mixed Systems* [113, 117]: This chapter extends and details the notion of trust with social and behavioral aspects. We demonstrate the applicability of social trust by simulating an example scenario where interactions between community members are balanced through the means of delegations.
  - *Chapter 5: Trusted Interaction Patterns in Enterprise Service Networks* [116]: We refine our delegation concepts, and discuss the interplay of interaction patterns and trust. For that purpose, we formulate our trust model and show its application in context of the Expert Web use case.

- Part II: Trust Mining and Prediction on the Web
  - *Chapter 6: Trust and Reputation Mining in Professional Virtual Communities* [119]: In the case there are no observable SOAP interactions, trust is going to be based on other data, gathered through mining on the Web. We present an approach to determining relations of users in online discussion forums.

- *Chapter 7: Bootstrapping and Prediction of Trust* [112]: This chapter deals with bootstrapping, i.e., predicting, trust between actors in the case no interactions from previous collaborations have been captured.

- Part III: Trust-based Service-centric Applications
  - *Chapter 8: Context-aware Interaction Models in Virtual Organizations* [118]: We go one step further towards the realization of the Expert Web use case, and introduce fundamental concepts to allow flexible expert discovery and involvements in process-oriented environments.
  - *Chapter 9: Trust-based Discovery and Interactions in Expert Networks* [105]: We deal with an advanced expert discovery approach in the previously introduced Expert Web. For that purpose, we extend Kleinberg's popular HITS algorithm with context-sensitive personalized trust weightings.
  - *Chapter 10: Trust-based Adaptations in Complex Service Systems* [115]: We discuss the meaning of trust for dynamic adaptations in service-centric systems. For that purpose, we highlight mechanisms for flexible information disclosure in collaboration scenarios.
  - *Chapter 11: Supporting Network Formation under Privacy Constraints* [114]: This chapter deals with privacy issues of utilizing data mining for trust inference and profile sharing for group formations in a Web of Social Trust. We discuss privacy concerns and investigate new models for flexibly sharing personal data in collaborative environments.

Finally, Chapter 12 concludes the paper. The appendix contains implementation details of the Collaborative Network Provider, a central component of several of our software frameworks presented in this thesis.

CHAPTER 2

Related Work

Contents

This dissertation deals with the meaning of trust in service-oriented mixed systems collaborations; starting with trust inference based on actor behavior, temporal evaluation of trust relations, the bootstrapping problem, and finally, the wide variety of trust-aware applications.

We structure our discussion on related work in the following sections:

- *Flexible and Context-aware Collaborations* describes the fundamental collaboration environment and related concepts to model context.

- *Interactions in Mixed Service-oriented Systems* deals with interaction types, styles, and patterns that represent the basis for automatic trust inference.

- *Autonomic Service-oriented Computing* deals with run-time discovery, selection, and adaptations of services for optimizations.

- *Behavioral and Social Trust Models in SOA* discusses existing trust models in SOA, as well as from other domains, including the agent community.

- *Expertise Ranking and Reputation Models* highlights state of the art ranking approaches for the Web that can be adopted for our mixed systems environment.

- *Information Disclosure under Privacy Constraints* uses trust to enable privacy-aware data sharing applications, e.g., to support flexible collaborations or group formations.

2.1 Flexible and Context-aware Collaborations

In collaborations, activities are the means to capture the context in which human interactions take place. Activities describe the goal of a task, the participants, utilized resources, and temporal constraints. People interact in the context of activities to successfully accomplish their goals. Studies regarding activities in various work settings are described in [48]. They identify patterns of complex business activities, which are then used to derive relationships and activity patterns; see [87, 88]. The potential impact of activity-centric collaboration is highlighted with special focus on the value to individuals, teams, and enterprises. Studies on distributed teams focus on human performance and interactions [9, 20, 94], even in Enterprise 2.0 environments [15]. Caramba [28] organizes work items of individuals as activities that can be used to manage collaborations. For example, one can see the status of an activity, who contributed to an activity, documents created within a particular activity, etc. Based on log analysis, human interaction patterns can be extracted [29]. Prior work dealing with the management of cross-organizational processes can be found for instance in [53].

For the last years, context has been at the center of many research efforts. As a multi disciplinary domain, multiple definitions exist, most of them fitting just a certain focus. In computer science the definition given by Abowd et al. [3] is amongst the most adopted ones. To get an overview, Baldauf et al. [8] provide a survey on context models, techniques, frameworks and applications.

2.2 Interactions in Mixed Service-oriented Systems

Major software vendors have been working on standards addressing the lack of human interaction support in service-oriented systems. WS-HumanTask [71] and Bpel4People [1] were released to address the emergent need for human interactions in business processes. These standards specify languages to model human interactions, the lifecycle of human tasks, and generic role models. The relation of Bpel4People-related Web standards and resource patterns was discussed in [101]. Role-based access models (see [71] and [82]) are used to model responsibilities and potential task assignees in processes.

An example for a *mixed system* is a virtual organization (VO) using Web 2.0 technologies. A VO is a temporary alliance of organizations that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks [17]. Nowadays, virtual organizations are more and more realized with SOA concepts, regarding service discovery, service descriptions (WSDL), dynamic binding, and SOAP-based interactions. In such networks, humans may participate and provide services in a uniform way by utilizing the Human-Provided Services (HPS) framework [106]. In HPS, we also follow a Web services-based approach to support human interactions in a service-oriented manner. However, HPS and mentioned Bpel4People are complementary, and not competing approaches. HPSs are services that can be created by end-users, whereas Bpel4People defines concepts (e.g., role model, logical people groups) to model interactions in BPEL-based application scenarios.

2.3 Autonomic Service-oriented Computing

The problem of composition and adaptation is strongly related to organization and control. The autonomic computing paradigm [56] advocates self-* principles to reduce human intervention in configuring systems and services. An autonomic computing environment has the ability to manage itself and dynamically adapt to changes in accordance with objectives and strategies. Inspired by the principles of control systems, the autonomic computing paradigm aims at achieving dynamic adaption of the system based on the actual execution context [69].

Enhanced flexibility of complex systems is introduced by establishing a cycle that feeds back environmental conditions to allow the system to adapt its behavior. This MAPE cycle [56] is considered as one of the core mechanism to achieve adaptability through self-* properties. While autonomic computing allows for autonomous elements and applies these principles to distributed systems, current research efforts left the human element outside the loop. Based on the observed context of the environment, different adaptation strategies can be applied [24] to guide interactions between actors, the parameters of those strategies, and actions to prevent inefficient use of resources and disruptions. In the context of multi agent systems (MAS), self-configuring social techniques were introduced in [16]. A major challenge in adaptation and self-configuration is to dynamically find the most relevant adaptation parameter. Research relevant to this issue can be found in [133].

2.4 Behavioral and Social Trust Models for SOA

2.4.1 Fundamental Trust Models

Marsh [74] introduced trust as a computational concept, including a fundamental definition, a model and several related concepts impacting trust. Based on his work, various extended definitions and models have been developed. Some surveys on trust related to computer science have been performed [7, 44, 58, 75, 80], which outline common concepts of trust, clarify the terminology and describe the most popular models. From the many existing definitions of trust, those from [44, 90] describe that trust relies on previous interactions and collaboration encounters, which fits best to our highly flexible environment.

Context dependent trust was investigated by [7, 44, 58, 74]. Context-aware computing focusing modeling and sensing of context can be found in [3, 13, 70].

2.4.2 Trust in SOA

Recently, trust in social environments and service-oriented systems has become a very important research area. SOA-based infrastructures are typically distributed comprising a large number of available services and huge amounts of interaction logs. Therefore, trust in SOA has to be managed in an automatic manner. A trust management framework for service-oriented environments has been presented in [21, 66, 72], however, without considering particular application scenarios with human actors in SOA. Although several models define trust on interactions and behavior, and account for reputation and recommendation, there is hardly any case study about the application of these models in service-oriented networks. While various theoretically sound models have been developed in the last years, fundamental research questions, such as the technical grounding in SOA and the complexity of trust-aware context-sensitive data management in large-scale networks are still widely unaddressed.

In particular, in our work we construct trust models that are closely connected to and applied in the SOA domain. These trust models rely on mechanisms of SOA, such as SOAP-based interaction monitoring, well-described communication interfaces, and WS-enabled collaboration infrastructures. Other works in that domain, such as [66, 107, 124] disregard the human factor in large-scale networks.

2.4.3 Behavioral and Social Trust Models

Welser et al. [129] provides interesting studies about social roles in online discussion groups and participants' behaviors. Furthermore, Nonnecke et al. [91] and Meyer et al. [84] research the meaning of online communication and differences between traditional face-to face and threaded discussions. McLure-Wasko and Faraj [81] investigate the motivation for knowledge sharing in online platforms, and Rheingold [98] comprehensively examines the concept of virtual communities. The article [122] in *The Economist* draws the importance of trust in business communication, and shows that various factors which directly influence trust between individuals are based on communication.

While some works, including [18, 77], underline social aspects and subjective components of trust, others research the detection of attacks and malicious behavior using appropriate trust models [121]. Trust models highlight concepts from either an abstract perspective, i.e., not bound to specific scenarios or environments, or show their application in certain domains, such as behavior in agent networks [127] or service-oriented environments [79].

Depending on the environment, trust may rely on the outcome of previous interactions [12, 90, 117], **and** the similarity of interests and skills [38, 78, 112, 134]. Note, trust is not simply a synonym for *quality of service* (QoS). Instead, metrics expressing social behavior and influences are used in certain contexts. For instance, *reciprocity* [90] is a concept describing that humans tend to establish a balance between provided support and obtained benefit from collaboration partners. The application of trust relations in team formations and virtual organizations has been studied before, e.g., in [63, 85, 137]. Trust propagation models [46, 76, 123, 135] are intuitive methods to predict relations where no personal trust emerged; e.g., transitive recommendations. Since trust propagation provides meaningful results, we propagate personal profile data over these trust relations.

In Chapter 4 we describe an approach to trust inference that is based on fuzzy set theories. This technique has been applied in trust models before [45, 68, 96, 109], however, to our best knowledge, not to interpret diverse sets of interaction metrics. Utilizing interaction metrics, in particular calculated between pairs of network members, enables us to incorporate a personalized and social perspective. For instance, an actor's behavior may vary toward different network members. This aspect is usually out of scope in Web Services trust models, that are often closely connected to traditional QoS approaches. Moreover, most trust models in the SOA domain utilize trust for service selection only (for instance see [79]), and neglect the collaborative aspects and the human factor.

2.4.4 Bootstrapping of Trust

Bootstrapping addresses the *cold start problem* and refers to putting a system into operation. Trust –from our perspective – cannot be negotiated or defined in advance. It rather emerges upon interactions and behavior of actors and thus, needs a certain time span to be built. However, until enough data has been collected, interests and skills can be used to predict potentially emerging trust relations. Mining, processing, and comparing user profiles is a key concept [38, 134]. In Chapter 7, we utilize tagging profiles of users to compare their centers of interests.

Tagging and its meaning has been widely studied in [40]. Several approaches have been introduced, dealing with the construction of hierarchical structures of tags [32, 52], generating user profiles based on collaborative tagging [86, 108], and collaborative filtering in general [50]. However, determining tagging profile similarities has not been addressed well in previous works. Therefore, we apply the concepts of tailored tagging profiles, and indirect similarity measurement. Our approach uses various mathematical methods from the domain of information retrieval, including term-frequency and inverse document frequency metrics [103], measuring similarities, and hierarchical clustering [99].

2.5 Expert Ranking and Reputation Models

We show in this work that models and algorithms to determine the expertise of users are important in future service-oriented environments. The notion of service-orientation is not only applicable to Web services. Service-orientation in human collaboration is becoming increasingly important. For example, task-based platforms allow users to share their expertise [130]; or users offer their expertise by helping other users in forums or answer communities [5, 59]. By analyzing email conversations, [27] studied graph-based algorithms such as HITS [65] and PageRank [93] to estimate the expertise of users. The authors in [110] followed a graph-entropy model to measure the importance of users. In [62], an email analysis in enterprises, defining information flow metrics in the social interaction graph was presented. The work by [132] followed a graph-based approach and applied HITS as well as PageRank in online communities (i.e., a Java question and answer forum). TrustRank [47] is one further approach to combine common ranking technologies and personalized trust.

While the above cited works attempted to model the importance of users based on interactions and information flows; they ignore the fact that interactions typically take place in different *contexts*. Approaches for calculating personalized PageRank scores [49, 57] were introduced to enable topic-sensitive search on the Web, but have not been applied to human interaction analysis. In contrast, we propose a model where expertise analysis is performed considering context information. Furthermore, we propose algorithms that can be computed online, while most other approaches demand for offline calculation due to computational complexity.

Reputation expresses the overall standing of community members and can be used to rank them in terms of diverse criteria. Reputation may be built by a combination of personal trust and further attributes, such as compliance [60], and formal constraints. Multi-criteria decision making and ranking, as surveyed in [35], is used to identify 'best' available collaboration partners (experts) when accounting for several input metrics. In Chapter 8, we make particularly use of the *Promethee* approach [14]. Trust relying besides explicit ratings, on monitoring and analyzing interactions and behavior of actors, can be one criterion to rank collaboration partners in social networks. The application of trust relations in team formations and virtual organizations has been studied in [63, 116, 137].

In Chapter 6, we interpret previous postings in online forums as interactions between people and rank them according to their trustworthiness in the originating network. There are several graph based ranking algorithms, including PageRank [93], HITS [65], and Eigentrust [61]. However, in contrast to these algorithms, which operate on top of an existing network, our approach tackles the challenges beneath, i.e. creating the interaction network based on discussion data. To this end, we develop a mining algorithm to gather individual trust relationships based on observed communications, considering detailed analysis of online discussion platforms such as Gomez et al. [42] for Slashdot and Adamic et al. [4] for Yahoo Answers.

2.6 Information Disclosure under Privacy Constraints

The interplay of trust and privacy has been studied in the areas of social networks [30] and electronic commerce [83]. Especially the latter work concludes that trust is strongly related to information disclosure, and thus, privacy. As users increasingly disseminate their information on the Web, privacy concerns demand flexible information access control mechanisms [89]. In some recent articles on the Web, e.g., see [31, 64], the authors discuss to what extent shared personal information of (naive) users may be exploited. Marsh discusses in an article [73] how trust models enhance information sharing among community members. Knowledge sharing behavior in virtual communities and the relation to trust has been investigated in [54].

There exist several works in the area of recommender systems (e.g., [125]) that use personal trust to optimize item recommendation (that is usually based on collaborative filtering only). One of the more related use cases is the recommendation of documents [51]. However, while they use trust to improve document recommendations (e.g., to better match interests of users), we restrict access based on context-aware personal relations. Others focus on traditional trust-based access control mechanisms [11] that are based on more static role models and relations.

The technical realization of *trusted information sharing* in the introduced science collaboration network in Chapter 10 is related to *selective dissemination of information* (SDI) [6, 25]. SDI deals with questions regarding which (parts of) data are shared with others, and mechanisms to disseminate data. We adopted concepts of SDI, such as the representation of information through XML, or mechanisms to process XML-based data.

Part I

Trust and Reputation Models in SOA

CHAPTER 3

The Cycle of Trust in Mixed Service-oriented Systems

Outline. This work starts with establishing a detailed understanding of the notion of trust in mixed service-oriented systems, including a first technical grounding in SOA. We discuss a fundamental approach to dynamically determine trust from the system's point of view, and further outline a user perspective.

Contents

3.1 Motivation

Service-oriented architectures (SOA) were understood as environments where software services are registered, discovered, and invoked. This limited perspective on SOA has changed over the past years because it has been realized by researchers and industry players that humans must be part of such systems. Not only interactions between software services and the ability to compose complex flows are important, but rather the interplay between compositions of human and software services. We define such environments as *mixed systems* comprising software services and Human-Provided Services (HPS, see [106]).

A key requirement for the success of mixed systems is the ability to support interactions in a flexible yet reusable manner. In this work, we focus on an activity-centric approach to support and manage interactions between various actors including people and software services. In contrast to many existing approaches based on workflow systems, activities are not predefined, for example by an administrator that is responsible for the design of a workflow. Activities can be modeled as templates before collaborations and interactions start; however, activities are flexible and can be reorganized by adding, removing, or delegating activities at run-time. Hence, activities are essential to allow for dynamics in interactions and to structure collaborations establishing the link between artifacts, services, and people. Existing work [22, 28, 87] has studied activities as a concept to cope with the challenges of dynamic collaboration environments.

Activity-centric collaboration is useful in many different application domains ranging from business-centric collaboration to large-scale social networking platforms. In enterprises, an activity management platform can be used to deal with the increasing complexity in coordination and communication. For example, an activity workspace allows people to manage their interactions (status, task progress, or artifact changes) as well as various services that can be used in the context of activities (e.g., communication services). On the other hand, people using social networking platforms might use activities in a much more informal way. Activities in a social context may depict ‘simple’ structures and flows, which are in a manner similar to user defined data or mashups of services. Such activity-based templates can be shared with other users to support collaborations.

We strongly believe that trust and reputation mechanisms are key to the success of open dynamic service-oriented environments. However, trust between human and software services is emerging based on interactions. Interactions, for example, may be categorized in terms of success (e.g., failed or finished). Therefore, an important aspect of our approach is the monitoring and analysis of interactions to automatically determine trust in mixed systems.

In this chapter, we present the following key contributions:

- *Definition of Trust.* We establish a detailed understanding of trust in mixed service-oriented systems, and required basic concepts, including flexible collaborations, types of interactions, and the role of context. Besides direct trust, which is the focus of this work, we deal with related concepts, such as recommendations and reputation too.

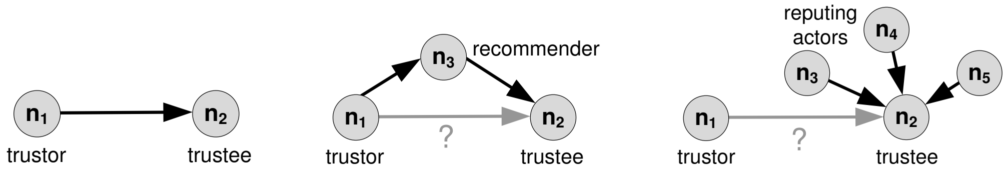
- *Fundamental Approach of Trust Determination.* We present our approach for the context-aware analysis of interactions to establish trust. Our approach relies on concepts from the autonomic computing domain, such as feedback loops and adaptation techniques.

- *Algorithm and Application.* We construct a graph-based algorithm to perform trust mining over heterogeneous sources of captured interaction data. The Vienna Trust Emergence Framework (VieTE) – first published in [120] – comprises the most important features for the determination and management of trust.

3.2 Trust Modeling in Collaborations

3.2.1 Actor Roles and Trust Relations

As common in the trust research literature, the roles of actors in a directed trust relation are defined as *trustor* (also *truster*), which is the trusting entity, and *trustee* which is the trusted entity [19]. Trust relations E between entities N – the actors in our mixed systems environment – are managed in a directed graph model $G = (N, E)$.



(a) Direct trust. (b) Recommendation. (c) Reputation.

Figure 3.1: Trust related concepts.

We distinguish between different kinds of trust-related social relationships in a network of collaborating entities.

Direct Trust Relations. These relations base on first-hand experiences and are inferred from the success and outcome of previous interactions between the trustor and the trustee (Figure 3.1(a)).

Recommendations. These relations, based on second-hand experiences, are inferred from the success and outcome of previous interactions between a well trusted entity and the trustee. This case is depicted in Figure 3.1(b), where the relation from n_1 to n_2 is derived by considering the relations from n_1 to n_3 and from n_3 to n_2, ultimately n_3 recommends n_2 to n_1.

Reputation. Reputation is a concept where trust of the trustor to the trustee is completely inferred from third party relationships as depicted in Figure 3.1(c). By considering trust of n_4, n_5, and n_6 in n_2, n_1 may derive a notion of trust in n_2. Even though reputation is the most unreliable substitute for trust (compared to first-hand experiences or recommendations), it is nevertheless a useful concept, especially the more third party relations to the trustee are considered.

3.2.2 Context of Trust

Trust is context dependent, which means trust relations cannot be determined generally, but with respect to a particular situation [7, 19, 58]. Generally each situation, which is determined and described by context data, is unique and consists of multi-dimensional properties.

We developed a system for trust inference in collaborative environments; thus we are able to reduce the complexity of context elements. We define that collaboration situations are basically reflected by describing the actually performed activity, including the goal and nature of work. In a simplified case, when considering only the type of activities performed, a human trustor might trust a trustee to *organize a meeting* (context 1), but not to *develop a software module* (context 2). In this case contexts are obviously different and not closely related to each other. However, there are cases where contexts may be similar. For instance, the trustor might have established trust to the trustee regarding *software implementation*. This means the trustor can trust the trustee to perform assigned activities in the area of *software implementation* reliably in the given time with the required quality. If the trustor wants to know how much s/he can trust the trustee with respect to *software testing* activities, trust can be inferred from the relation regarding *software implementation*, because both *implementation* and *testing* are part of software development, and thus activities in both fields will typically have similar requirements. Hence, the concept of trust context allows (i) distinguishing trust relations with respect to different contexts and thus, expressing trust relations more precisely and reliably, and (ii) deriving relations for new situations based on relations in other, but similar situations. We call the second case *trust propagation over contexts*. This concept permits considering trust relations established in different contexts by taking the similarities between contexts into account.

3.2.3 The Diversity of Trust

Trust is a diverse concept and relies on various impacting factors. Figure 3.2 depicts influences, primarily discussed in the literature, on the trust relationship from a *trustor* to a *trustee* with respect to *context 1*.

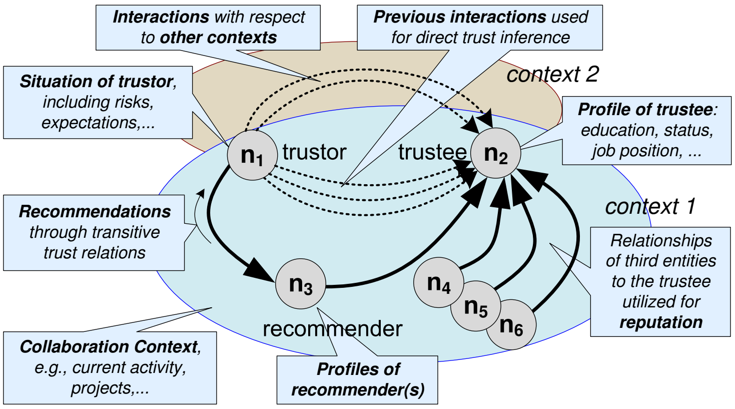

Interactions with respect to other contexts
Previous interactions used for direct trust inference
Situation of trustor, including risks, expectations,...
context 2
Profile of trustee: education, status, job position, ...
n1
trustor
trustee
n2
Recommendations through transitive trust relations
context 1
Relationships of third entities to the trustee utilized for reputation
n3
n4
n5
n6
recommender
Collaboration Context, e.g., current activity, projects,...
Profiles of recommender(s)

Figure 3.2: Factors influencing trust.

The *profile of the trustee*, such as the educational status and job position, is utilized, e.g., when it comes to collaboration partner selection or activity assignment. The profile describes if the trustee owns the formal competencies to be trusted to perform a given activity reliably.

The current *situation of the trustor*, such as his risks and expectations with respect to a particular situation is a further influencing factor. Decisions about whom to trust always depend on the risks the trustor is facing. If the trustor wants to assign the trustee a particular activity, but the negative consequences for the trustor are high in case the activity is not performed well, then the trustor will not trust the trustee carelessly.

The *collaboration context*, describing the activity to perform or the overall project to realize, is used to determine which *previous interactions* have to be aggregated to determine the direct trust relation. *Interactions with respect to other contexts*, can be utilized for trust inference as well (however with reduced relevance), especially if previous situations and current situation share similar contextual properties.

Furthermore, direct *recommendations* from well-known and trusted recommenders are taken into account. This concept utilizes the transitivity of trust and is known as trust propagation. Experiments have shown that this concept works well in several cases [46, 77]. The *profiles of recommenders*, may represent a valuable source of data to decide if and how much someone's recommendation can be trusted.

The *reputation of the trustee*, is defined as the aggregated community trust and determined by the sum of all trust relations from different entities in the trustee. Considering reputation only is the most unreliable kind of determining trust, because it does not take personal preferences into account. However, even if the trustor has not established relations to the recommending entities, a reasonable notion of trust can be inferred when relying on large sets of opinions.

3.3 From Interactions to the Cycle of Trust

Before we show our new approach for dynamic trust inference, we introduce the most important applied concepts.

3.3.1 Supported Interactions

We define a *mixed systems environment* which consists of actors of different types, including humans, services, and also humans offering their capabilities as services (Human-Provided Services [106]), interacting with each other to perform a given set of activities contributing towards a common goal. In this environment, we distinguish between the following interactions : (a) *Human-service interactions:* services can be used to realize collaboration functions in a flexible manner. Humans are able to use services (the representation frontend) to perform collaborations. Typical human-service interactions include map services, document sharing service, etc. (b) *Human interactions as part of software service compositions:* many service interaction scenarios demand for human interactions. A popular example is BPEL4People [1]. (c) *Human interactions using Human-provided Services:* Our previous work included the support of service-human interactions (e.g., see [106]), allowing humans to express and offer their capabilities as services. (d) *Interactions between software services:* such interaction scenarios are found in compositions of software services. For example, output of service A is used as input by service B. (e) *Service initiated interactions towards humans:* such scenarios include notifications or news feeds, which are pushed toward humans.

The context of interactions, e.g., their purpose and intention, has to be captured to enable meaningful trust determination. While the context of unstructured interactions is normally manifold, we argue that with certain assumptions, interactions can be appropriately categorized. Imagine an online collaboration environment, where people collaboratively define and perform activities of certain types. Typed interactions within that activities, such as support requests, or delegations, can be rated in terms of success and impact on ongoing work.

3.3.2 Interaction Context

Our system is based on the notion of activities. Activities represent work done and actions performed by actors of different type. The activity model, as shown in Figure 3.3, allows collaborations to be structured in a flexible way.

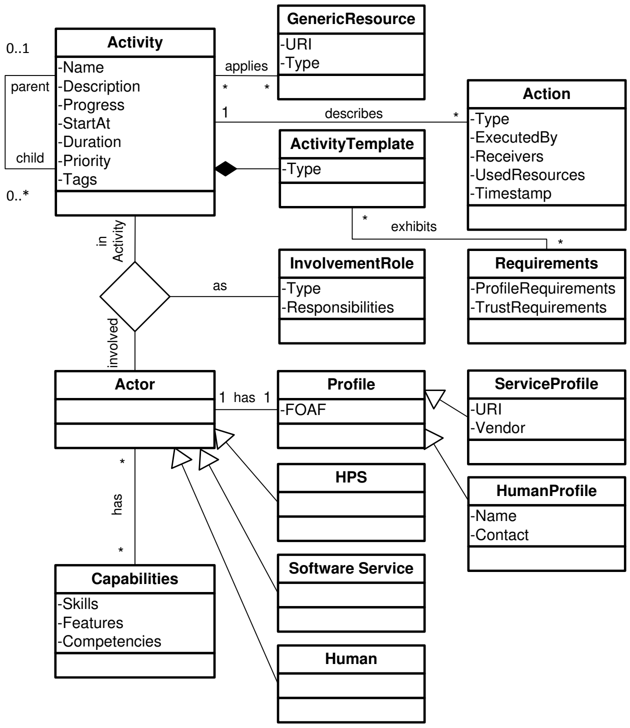


Figure 3.3: Activity model.

The model supports dynamic collaborations in mixed systems because activities can be created at runtime. This is needed to support more flexibility and agility in interactions. An activity may comprise a set of sub-activities which can be assigned to different actors. Activity templates describe the ’how-to’ execute a particular activity and what services to be used to work on that activity, and are associated with requirements that are used to control the assignment of actors. Activities can be organized in sets to assemble more complex activities out of the simple ones. The execution of activities and services has to be based on the context of the environment. In activity-centric collaboration, context is any information that lets the actor understand and estimate the current situation, work environment, and the information that is relevant given the actor’s current activity. For example, an actor working on an activity needs to know details such as the description, progress and start date of the activity, associated artifacts and their status; information regarding tools, services, and available devices; and context details regarding co-actors (presence status or location information), and finally discussions that are relevant for an activity. However, services are not only used as resources to accomplish activities; instead services play an active role in compositions. For example, services can trigger notifications or reminders based on context information.

During the execution activities can also be delegated to change the ownership and responsibility for a specific activity. Actors perform different types of actions, such as coordinating work, discussing with other actors, or executing some kinds of tasks. The combination of executed actions and performed activities appropriately reflects the situation of interacting actors for trust inference.

3.3.3 The Cycle of Trust

Our system design follows the MAPE approach [56], as depicted in Figure 3.4. MAPE, incorporating basic concepts from the control engineering domain, describes a cycle consisting of four phases, which are *monitor*, *analyze*, *plan* and *execute*. Periodically running through these four phases establishes a kind of environmental feedback control, and therefore, allows to adapt to varying circumstances. Applied in our environment, we are able to infer trust dynamically during ongoing collaborations.

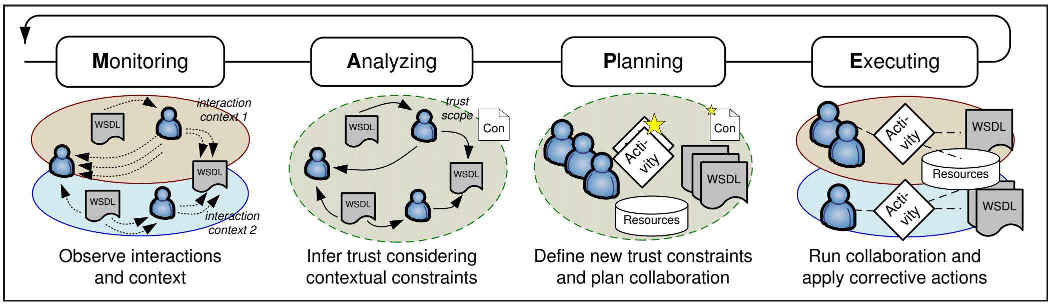

Monitoring
Analyzing
Planning
Executing
WSDL
interaction context 1
interaction context 2
trust scope
Con
Acti-vity
Resources
Observe interactions and context
Infer trust considering contextual constraints
Define new trust constraints and plan collaboration
Run collaboration and apply corrective actions

Figure 3.4: The MAPE cycle applied for dynamic trust inference.

In the **Monitoring Phase** our system observes ongoing collaborations, particularly the creation of new activities, actions executed therein, and interactions taking place, according to the previous descriptions, between humans, services, and HPSs. These interactions, including their types and contexts, are captured and modeled as an interaction network utilized in further trust analyzes. In the **Analyzing Phase** the created interaction network is used to infer trust relationships. For this purpose, the relevance of each interaction is graded automatically considering configurable trust constraints (Con). These constraints define the *scope of trust* and depend on the purpose of trust inference, e.g., constraints differ when calculating trust in an actor to fulfill a particular activity type, or trust in an actor to hold a particular involvement role. Direct trust relationships are calculated by evaluating interactions and their relevance in the defined scope. Based on these direct trust relationships, the concepts of recommendation and reputation are applied to establish relationships relying on second-hand experience and collective opinions. The following **Planning Phase** covers the set up of collaboration scenarios taking the inferred trust relations into account.

Furthermore, trust constraints for the next run of the cycle are configured with respect to planned activity requirements. This means it is set up which contextual properties have to be taken into account when calculating trust. This depends on the planning use case, such as the composition of actors, the assignment of roles and responsibilities, the assignment and scheduling of activities, and sharing of artifacts. The **Execution Phase** provides support to enhance the execution of activities, including observing activity deadlines, solving activity scheduling conflicts, checking the availability of actors, and compensation of resource limitations. Furthermore, in parallel the collaboration behavior of actors is monitored, including their actions and interactions. This closes the cycle of trust.

3.4 Implementation of Fundamental Trust Inference in SOA

In this chapter, we outline the fundamental approach to automatic trust inference in SOA. This includes the monitoring of interactions, their rating, weighting, and aggregation to trust. A more advanced rule-based approach that interprets actor behavior is introduced in the next Chapter.

We implemented the VieTE architecture depicted in Figure 3.5, that has been developed utilizing the MAPE approach. Users, logged in to the management portal, can register new humans and services by entering their profile data. Furthermore, they manage own activities and specify trust constraint sets. On system level, interactions are monitored, captured by software sensors, and analyzed using the configured trust constraints. Recent analysis results lead to the formation of new trust relations, and impact existing ones. This knowledge about existing trust relations can be retrieved through a Web service component (Trust Provisioning). On the one side, the management portal itself uses this knowledge, e.g., to suggest new team compositions or activity delegations based on interpersonal trust. On the other side, further third-party collaboration tools could use this knowledge as it is provided in a standardized manner (SOAP and REST-based versions).

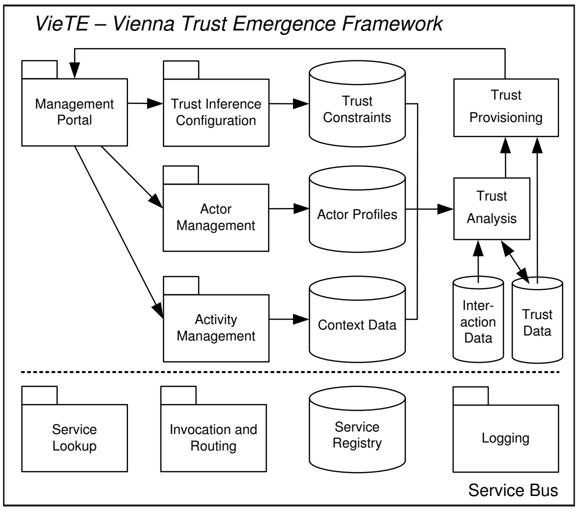


Figure 3.5: VieTE architecture.

3.4.1 Monitoring

The Logging component captures interactions during activity execution, such as human communication through services and Web service calls (Listing 3.1), e.g., through intercepting SOAP calls, and explicit actions undertaken by actors through VieTE's Management Portal, such as the delegation of activities (see Listing 3.2). Data of both sources are processed, including analysis of faults and interaction correlation to discover request-response patterns [41], and converted to more generic interactions. This generic type describes in detail the type and success of an interaction between two particular actors (Listing 3.3).

```
<ServiceInteraction xmlns="http://www.coin-ip.eu/ns/invocation">
  <clientEndpoint>192.168.0.101</clientEndpoint>
  <messageCorrelationID>000a1460-25ba-...</messageCorrelationID>
  <messageType>Response</messageType>
  <serviceEndpoint>http://www.coin-ip.eu/ss/IMService</serviceEndpoint>
  <eventSourceID>AL-invoke@192.168.0.100</eventSourceID>
  <timeStamp>1207212091812</timeStamp>
</ServiceInteraction>
```

Listing 3.1: Service interaction log example.

At the end of each run through the monitoring phase, an interaction network is built based on available generic interactions and stored in the interaction database (see Figure 3.5). We model this network as a directed graph $G_I = (N, E_I)$, whose nodes N represent the actors and multiple edges E_I reflect interactions between them. An edge $e_\imath = (n_i, n_j, \imath, ctx)$ is described by the source n_i of an interaction, the sink n_j, the generic interaction $\imath$ with its properties, and the interaction context ctx.

```
<Action xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.coin-ip.eu/ns/action"
   xsi:type="CoordinationAction"
   ActionURI="http://www.coin-ip.eu/CoordinationAction#6564"
   DescribesActivityURI="http://www.coin-ip.eu/Activity#222"
   Timestamp="2009-03-05T15:13:21.563Z">
   <ExecutedBy>http://www.coin-ip.eu/Actor#Daniel</ExecutedBy>
   <CoordinationType>
      <DelegateType>Delegate</DelegateType>
   </CoordinationType>
   <ToActor>http://www.coin-ip.eu/Actor#Florian</ToActor>
</Action>
```

Listing 3.2: Action log example.

```
<GenericInteraction xmlns="http://www.coin-ip.eu/ns/interaction">
   <sender>http://www.coin-ip.eu/Actor#Daniel</sender>
   <receiver>http://www.coin-ip.eu/Actor#Florian</receiver>
   <class>human-human</class>
   <type>Coordination</type>
   <subtype>MeetingOrganization</subtype>
   <numberOfConcernedCoActors>5</numberOfConcernedCoActors>
   <success>true</success>
   <successLevel>3</successLevel>
   <faultLevel>0</faultLevel>
   <faultReason></faultReason>
   <transportService>http://www.coin-ip.eu/ss/IMService</transportService>
   <context>http://www.coin-ip.eu/Activity#222</context>
   <timeStamp>1207212091812</timeStamp>
</GenericInteraction>
```

Listing 3.3: Generic interaction log example.

3.4.2 Analyzing

We understand trust to emerge from and to be highly impacted by the success (or faults) and types of interactions, considered with respect to particular situations described by the introduced context model.

Symbol	Description
α	impact factor of $\Delta\tau$ on τ
Con	set of constraints determining the relevance of $\imath$ in a scope
ctx	interaction context described by activity structures
$\Delta\tau$	trust obtained from recent interactions captured in the last run of the execution phase
$e_\imath$, e_t	edge in G_I and G_T respectively
f_I	fault score based on a set of $\imath$
γ	relevance of $\imath$ calculated using Con
G_I	interaction network modeled as graph $G_I = (N, E_I)$
G_T	trust network modeled as graph $G_T = (N, E_T)$
$\imath$	generic interaction between actors
n_i	source node of an interaction
n_j	sink node of an interaction
n_x	intermediate node between n_i and n_j
s_I	success score based on a set of $\imath$
τ	direct trust $\in [0,1]$ evolving over time
τ_{rec}	recommended trust from neighbors

Table 3.1: Symbol descriptions.

For inferring trust relations, we define a second graph model, the trust network $G_T = (N, E_T)$, where the actors N are the same as in G_I, but the edges $e_\tau = (n_i, n_j, \tau, Con)$ reflect the level of direct trust $\tau \in [0,1]$, emerging from interactions, with respect to the satisfaction of given constraints Con. Algorithm 1 is used to update G_T with the captured G_I, applying the given constraints Con used to weight the influence of interactions on trust calculation.

Algorithm 1 is periodically executed by the system to update the trust network, stored in the trust database (see Figure 3.5), considering captured interactions in pre-defined time intervals. The basic mode of operation is as follows: (i) Retrieve G_I built from recent interactions in the previous run of the execution phase. Furthermore get current G_T and configured constraints. (ii) Extract all interactions from G_I between the ordered pair of actors (n_i, n_j). (iii) Determine aggregated success score (s_I) and fault score (f_I) for available interactions from n_i to n_j. These scores are based on the level of success and fault respectively of an interaction $\imath$, and on the satisfaction of constraints with respect to the interaction context (γ in Equation 3.1 expresses the relevance of an interaction for given constraints). (iv) Calculate trust ($\Delta\tau$) from n_i to n_j only based on recent interactions. (v) Update previous trust τ with $\Delta\tau$ by applying the exponential moving average[1] method

[1] http://www.itl.nist.gov/div898/handbook/

Algorithm 1 Periodic update of G_T with recently captured G_I applying Con

```
/* retrieve $G_I = (N, E_I)$ from the interaction db */
/* retrieve $G_T = (N, E_T)$ from the trust db */
/* retrieve $Con$ from the constraints db (trust scope)*/
for each $n_i \in N$ do
    for each $n_j \in N$ do
        if $getEdges(E_I, n_i, n_j) \neq \emptyset$ then
            if $getEdge(E_T, n_i, n_j, Con) = 0$ then
                $e_t \leftarrow createEdge(n_i, n_j, 0, Con)$
            else
                $e_t \leftarrow getEdge(E_T, n_i, n_j, Con)$
            $f_I \leftarrow 0$, $s_I \leftarrow 0$
            $\tau \leftarrow getTrust(e_t)$
            for each $e_\imath \in getEdges(E_I, n_i, n_j)$ do
                $\imath \leftarrow getInteraction(e_\imath)$
                $ctx \leftarrow getContext(e_\imath)$
                $\gamma \leftarrow satisfy(Con, ctx)$
                $s_I \leftarrow s_I + successLevel(\imath) \cdot \gamma$
                $f_I \leftarrow f_I + faultLevel(\imath) \cdot \gamma$
            $\Delta\tau \leftarrow \frac{s_I}{s_I + f_I}$
            $\tau \leftarrow \Delta\tau \cdot \alpha + \tau \cdot (1 - \alpha)$
            $updateTrust(E_T, e_t, \tau)$
/* save updated $G_T$ in trust db */
/* dispose $G_I$ in interaction db */
```

using the weighting factor $\alpha \in [0, 1]$. (vi) Update changed trust edge e_t in G_T. (vii) Repeat steps (ii) to (vi) for each ordered pair of actors. (viii) Finally, save G_T, and dispose processed G_I.

The function $satisfy()$ applies Equation 3.1 to determine the relevance $\gamma \in [0, 1]$ of an interaction by comparing to which extend configured constraints Con match to the interaction context ctx. Each single constraint is set up with a weight. The result of the function $match()$ is either true or false.

$$\gamma = \frac{\sum_{\forall c \in Con} match(c, ctx) \cdot weight(c)}{\sum_{\forall c \in Con} weight(c)} \tag{3.1}$$

On top of G_T we realize algorithms for deriving recommendations and reputations. Algorithm 2 implements the inference of a collective recommendation trust τ_{rec} from a trustor n_i to a trustee n_j, by evaluating all second-hand experiences of the nodes $nodeList$ with respect to constraints in the given trust scope (Con not shown for the sake of brevity). Recommendations from different nodes $n_x \in nodeList$ may have different impact (e.g., depending on the actors' roles) reflected by $im(n_x)$. The concept of reputation can be realized in a similar way, but without accounting for the connections to a particular trustor.

The outcome of the analyzing phase is a global view on trust relations between actors.

Algorithm 2 recommendation($n_i, n_j, nodeList$)

```
1: $\tau_{rec} \leftarrow 0$
2: $sum \leftarrow 0$
3: for each $n_x \in nodeList$ do
4:     if $predecessor(n_x) = n_i \wedge successor(n_x) = n_j$ then
5:         $\tau_{rec} \leftarrow \tau_{rec} + \tau(n_i, n_j) \cdot \tau(n_x, n_j) \cdot im(n_x)$
6:         $sum \leftarrow sum + im(n_x)$
7: return $\frac{\tau_{rec}}{sum}$
```

The *Trust Provisioning* Web service interface (see Figure 3.5) offers the ability to query the periodically updated G_T using, for example, the *Management Portal*. Besides directed trust relations in G_T, recommendations as well as reputations with respect to configured constraints can be dynamically obtained.

3.4.3 Planning

Planning collaborations includes the selection of trusted actors for particular activities. Thus, defining the constraints used to infer trust from interactions is part of the planning phase. These constraints are configured for specific trust cases. As an example, consider that trust in humans has to be determined regarding their management skills. Constraints will be set up in order to parameterize the algorithm emphasizing the performance of past organizational activities and management interactions therein.

For defining constraints we integrated the popular Drools[2] engine in VieTE and utilize the Java semantic module. Listing 3.4 shows an example constraint definition. A set of such constraint rules build the definition of trust scopes, i.e., the situations and assumptions for which calculated trust is valid. Furthermore, the relevance of each constraint is weighted (ConWeight). This weight is used by the function $satisfy()$ to determine the degree to which constraints are fulfilled with respect to each interaction's context.

```
<rule-set name="trust_constraints"
   xmlns="http://drools.org/rules"
   xmlns:java="http://drools.org/semantics/java">
   <application-data identifier="results">
      java.util.HashMap
   </application-data>
   <rule name="CheckIfTypeOfActivityIsOrganizational">
      <parameter identifier="context">
         <class>at.ac.tuwien.infosys.viete.InteractionContext</class>
      </parameter>
      <java:condition>
         context.getActivity().getType().equals("organizational")
      </java:condition>
      <java:consequence>
         results.put("RuleActivityTypeIsOrg", ConWeight.MEDIUM);
      </java:consequence>
   </rule>
   <rule name="...">
      <!-- ... -->
   </rule>
</rule-set>
```

Listing 3.4: Constraint definition example.

[2] http://sourceforge.net/projects/drools/

3.4.4 Executing

In the execution phase the actual collaboration between actors in activities takes place. Every collaboration system requires typical procedures such as escalations that are triggered based on missed deadlines or limited resource availability. VieTE supports these procedures by providing trust relations between affected actors and thus supporting decision making to solve the problems.

3.5 Discussion

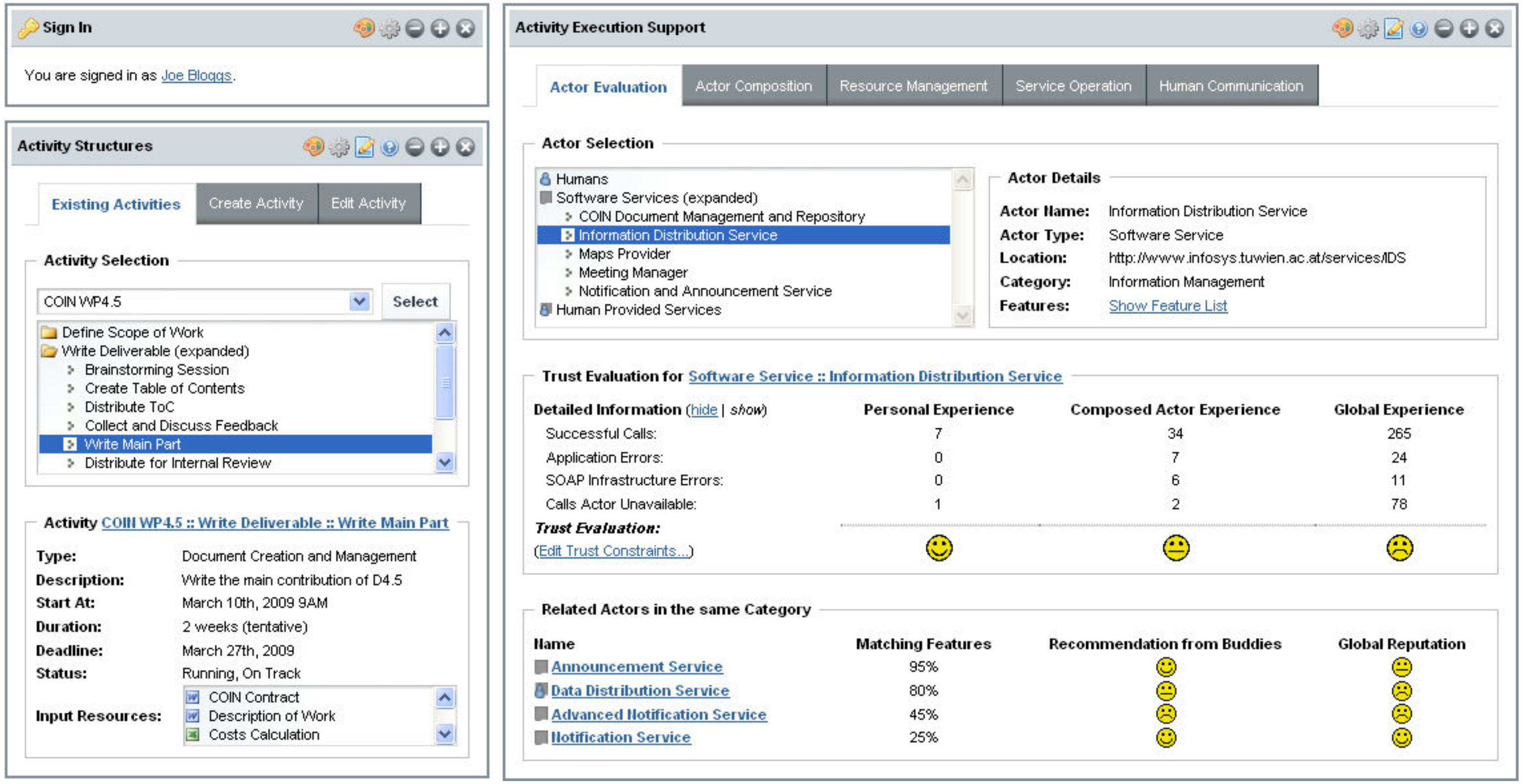
In this section we present an example screenshot of VieTE's *Management Portal* in Figure 3.6 to demonstrate the application of the VieTE framework in a real world collaboration scenario. We show how the end-user is supported in trust-based selection of actors to perform a specific activity.

In the left frame (Activity Selection) an activity structure is visualized. The details of the selected activity are shown in the lower box, including the name and type, a short description, temporal constraints (deadlines), the current status (pending, running, paused, finished, failed), and assigned resources (e.g., documents).

The right frame (Activity Execution Support) consists of 5 tabs:

- *Actor Evaluation* providing information about the user's personal trust relations to other actors as well as their reputation.
- *Actor Composition* is used for creating new 'mixed' teams, i.e., compositions of humans and services.
- *Resource Management* enables users to manage virtual resources, such as documents, and physical resources, including conference room reservations.
- *Service Operation* provides facilities to dynamically interact with software services by generating custom user interfaces and SOAP messages based on services' operations.
- *Human Communication* provides facilities to dynamically interact with humans by the means of e-mail, instant messaging or text chats.

We assume that the user has certain trust preferences, for example, selecting the most trusted service. (However, at this stage we do not consider trade-off models to account for multiple criteria such as costs versus trust.) Therefore, the top box of the right frame allows the selection of a particular actor to be evaluated. The results of trust evaluation in this co-actor (expressed as emoticons: happy, neutral, sad), based on interaction metrics such as successful calls and availability, is visualized. It is shown that the 'Information Distribution Service' behaves trustworthy for the logged in user (personal evaluation: happy emoticon), however the composed actor experience from the involved activity members is only medium (neutral emoticon). The global experience is largely negative (sad emoticon).



You are signed in as Joe Bloggs.

Activity Selection

COIN WP4.5

- Define Scope of Work
- Write Deliverable (expanded)
  - Brainstorming Session
  - Create Table of Contents
  - Distribute ToC
  - Collect and Discuss Feedback
  - Write Main Part
  - Distribute for Internal Review

Activity COIN WP4.5 :: Write Deliverable :: Write Main Part

Type:	Document Creation and Management
Description:	Write the main contribution of D4.5
Start At:	March 10th, 2009 9AM
Duration:	2 weeks (tentative)
Deadline:	March 27th, 2009
Status:	Running, On Track
Input Resources:	COIN Contract Description of Work Costs Calculation

Actor Selection

- Humans
- Software Services (expanded)
  - COIN Document Management and Repository
  - Information Distribution Service
  - Maps Provider
  - Meeting Manager
  - Notification and Announcement Service
- Human Provided Services

Actor Details

Actor Name:	Information Distribution Service
Actor Type:	Software Service
Location:	http://www.infosys.tuwien.ac.at/services/IDS
Category:	Information Management
Features:	Show Feature List

Trust Evaluation for Software Service :: Information Distribution Service

Detailed Information (hide | *show*)	**Personal Experience**	**Composed Actor Experience**	**Global Experience**
Successful Calls:	7	34	265
Application Errors:	0	7	24
SOAP Infrastructure Errors:	0	6	11
Calls Actor Unavailable:	1	2	78
Trust Evaluation: (Edit Trust Constraints...)	🙂	😐	🙁

Related Actors in the same Category

Name	**Matching Features**	**Recommendation from Buddies**	**Global Reputation**
Announcement Service	95%	🙂	😐
Data Distribution Service	80%	😐	🙁
Advanced Notification Service	45%	🙁	🙁
Notification Service	25%	🙂	🙂

Figure 3.6: VieTE trust and activity management portal.

The lower box shows actors with similar features as the currently selected one, for supporting fast replaceability (here: three software services and one Human-Provided service). Furthermore, their recommendations from well-trusted actors (‘buddies’) and their global reputation is visualized.

CHAPTER 4

Trustworthy Interaction Balancing in Mixed Systems

Outline. This chapter extends and details the notion of trust with social and behavioral aspects. We demonstrate the applicability of social trust by simulating an example scenario where interactions between community members are balanced through the means of delegations.

Contents

4.1 Motivation

Several works have previously shown [7, 44, 58] that trust and reputation mechanisms are key to the success of open dynamic service-oriented environments. However, trust between human and software services is emerging based on interactions. Interactions, for example, may be categorized in terms of success (e.g., failed or finished) and importance. Therefore, a key aspect of our approach is the monitoring and analysis of interactions to automatically determine trust in mixed service-oriented systems. We argue that in large-scale SOA-based systems, only automatic trust determination is feasible. In particular, manually assigned ratings are time-intensive and suffer from several drawbacks, such as unfairness, discrimination or low incentives for humans to provide trust ratings. Moreover, in mentioned mixed systems, software services demand for mechanisms to determine trust relations to other services.

In this chapter, we extend the notion and concepts of trust with social aspects, and present an approach to infer trust through rule-based interpretation of interaction metrics. While we highlighted the overall *Cycle of Trust* in the previous chapter, we focus now particularly on trust inference itself.

We adopt common definitions of trust in collaboration environments [44, 90, 111] and define *social trust* – also referred to as *trust* only for the sake of brevity – as follows:

> *Social Trust reflects the expectation one actor has about another's future behavior to perform given activities dependably, securely, and reliably based on experiences collected from previous interactions.*

This definition implies several key characteristics that need to be supported by a foundational trust model:

- Trust reflects an expectation and, therefore, cannot be expressed objectively. It is influenced by subjective perceptions of the involved actors.
- Trust is context dependent and is basically valid within a particular scope only, such as the type of an activity or the membership in a certain team.
- Trust relies on previous interactions, i.e., from previous behavior a prediction for the future is inferred.

Much research effort has been spent on defining and formalizing trust models (e.g., see [2, 55, 90]). We present the following novel contributions:

- *Interaction-based Trust Emergence.* We show the establishment of trust based on dynamic interaction patterns [29] in mixed service-oriented environments.
- *Trust Interpretation.* Due to the dynamic aspects in mixed systems, we do not apply a 'hard-wired' and static analytical trust model, but consider the subjective view and nature of trust by applying a rule-based interpretative approach.
- *Implementation and Architecture.* We realize our theoretical concepts with existing Web service standards, accounting for and relying on mechanisms that are typically available in service-oriented systems, such as WSDL service descriptions, and logging of SOAP-based interactions.

4.2 Interactions and Compositions

We depict a professional virtual community (PVC) environment to familiarize with our concepts, and to demonstrate the emergence of trust. A PVC is a virtual community that consists of professionals and experts who interact and collaborate by the means of information and communication technologies to perform their work. Nowadays, service-oriented technologies are used to realize PVCs. The actors, i.e., the community network members, that are both humans and software services, provide *help and support* on requests

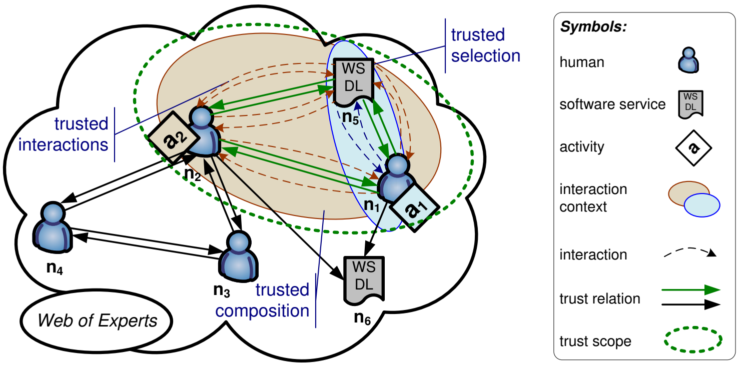


Figure 4.1: A mixed service-oriented PVC.

of each other. In such a mixed service-oriented environment actors have to register at a central community management service to become part of the network. Humans can register themselves by providing their profiles, including their education, employment status, certified skills and project experience. Services can be registered by their vendors or third party persons that offer information about service features and capabilities.

In the described environment, collaboration links are managed in a graph model $G = (N, E)$. Network members N are connected through social and organizational links E. These actors of the mixed system perform activities collaboratively. Activities are a concept to structure information in ad-hoc collaboration environments, including the goal of the ongoing tasks, involved actors, and utilized resources. They are either assigned from outside the community, e.g. belonging to a higher-level process, or emerge by identifying collaboration opportunities. Such an opportunity is for instance writing a research paper because of having required skills, and knowing *and trusting* the right supporting actors in a community (i.e., humans with the required knowledge, services providing scientific data).

In the scenario depicted by Figure 4.1, the two humans n_1 and n_2 are the owners of activities a_1 and a_2 respectively. We assume activity a_1 is a software implementation activity and a_2 is a software testing activity in some higher-level software development process (not depicted here). The human n_1, requests support from the Web service n_5, that is a software implementation knowledge base, providing code examples and FAQs[1] about software implementation. The dashed arrows represent interactions (requests for support (RFSs)), such as retrieving articles from the knowledge base. Interactions are performed by traditional SOAP calls. Even the capabilities of humans are described by WSDL and communication takes place with SOAP messages (see Human-Provided Services [106]). The interaction context, described by activity a_1 (reflected by the blue-shaded area), holds information about involved actors, goal of the activity, temporal constraints (start, duration, milestones), assigned resources, planned costs, risk with respect to the whole software development process and so on. The detailed description is not in scope of this work, however, we conclude, that an activity holistically describes the context of an interaction in our environment model [111].

[1] Frequently Asked Questions

Human n_2, the owner of activity a_2, performs his/her activity (software testing) jointly with the help of n_1 and n_5. For that purpose, s/he interacts with all activity participants, such as requesting help and assigning sub-activities. As defined before, trust emerges from interactions, and is bound to a particular scope. Therefore, we aggregate interactions that occurred in a pre-defined scope, calculate metrics (numeric values describing prior interaction behavior), and interpret them to establish trust. The scope of trust is reflected by the green dashed ellipse in Figure 4.1. In the given scenario, the scope comprises trust relations between PVC members regarding help and support in 'software development'. So, regardless of whether interactions took place in context of activity a_1 or a_2, interactions of both contexts are aggregated to calculate metrics, because both interaction contexts adhere to the scope of software development. Finally, interaction metrics are interpreted using rules, and the degree of trust between each pair of previously interacting members is determined.

Let us assume we are able to infer meaningful trust relations between interacting network members (as demonstrated later in this work). Usually, once a network member becomes highly trusted by others (normally leading to globally high reputation), s/he is consulted in future collaborations again and again. Hence, distinguished experts would get overloaded with work and flooded with support requests. We aim at applying a balancing model that relies on the means of delegations. For instance, if network member n_2 is overloaded, s/he may delegate incoming requests (e.g., from n_1) to third, well trusted, network members (e.g., n_5) in the same scope. These third parties may directly respond to the original requester. The delegation model has two important properties:

- *Interaction Balancing.* Interactions are not focused on highly reputed members only, but load is distributed over the whole network.

- *Establishment of new Trust Relations.* New personal trust relations that rely on direct interactions, emerge, leading to future trustworthy compositions.

4.3 Interpretative Trust Inference

In this dissertation, we gradually develop and extend the *VieTE - Vienna Trust Emergence Framework* [111] to research and evaluate novel concepts of trust and reputation in mixed service-oriented system environments. Briefly (see Figure 4.2), the system captures interactions between network members (bottom layer), calculates metrics of member relations, such as average response time, request success rates, and availability, performs a rule-based interpretation of these metrics, and infers trust between each pair of interacting members (middle layer). Finally a social network, describing collaboration- and trust relationships is provided (top layer). While the depicted architecture follows a centralized approach, the logging facilities are replicated for scalability reasons, and monitoring takes place in a distributed manner. Interactions are purged in predefined time intervals, depending on the required depth of history needed by metric calculation plugins.

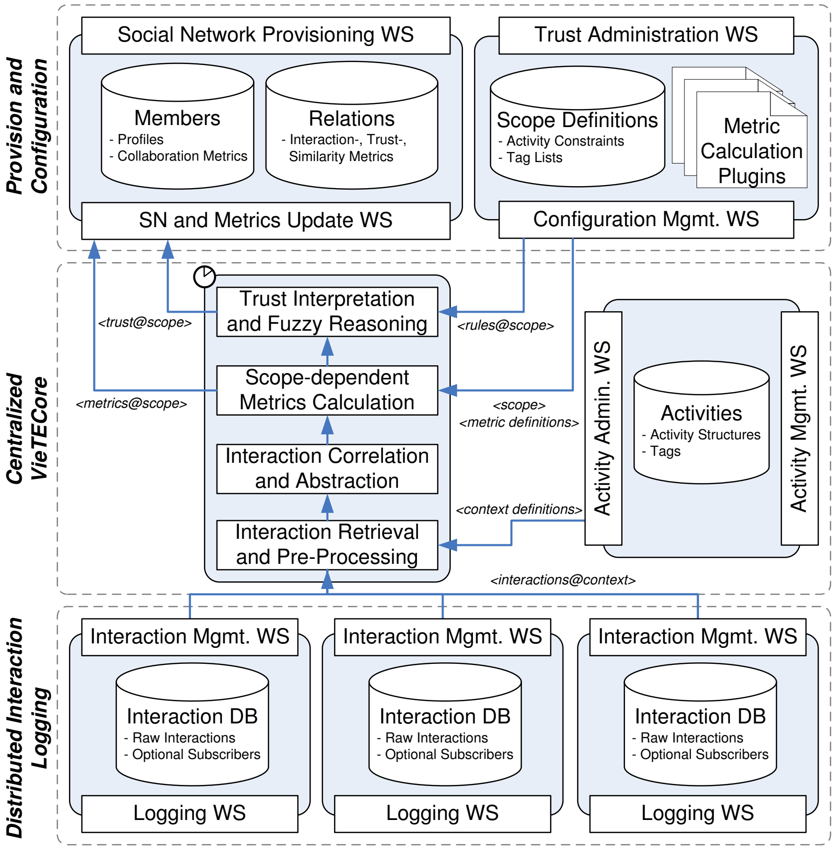


Figure 4.2: VieTE framework overview.

4.3.1 Interaction Monitoring

Interactions are captured by interaction sensors, stored and managed by logging services. The requests for support (RFSs) and their responses, exchanged between community members, are modeled as traditional SOAP calls, but with various header extensions, as shown in Listing 4.1. These header extensions include the context of interactions (i.e., the activity that is performed), delegation restrictions (e.g., the number of hops), identify the sender and receivers with WS-Addressing[2], and hold some meta-information about the RFS itself. For Human-Provided services (HPSs), SOAP messages are mapped to a GUI by the HPS framework [106].

Actors use activities to manage their work as introduced before. Activities are structures to describe work and its goals, as well as participating actors, used resources, and produced project artifacts. A detailed description of this model, used to capture the context of interactions, is provided in [111].

[2] http://www.w3.org/Submission/ws-addressing/

4.3.2 Interaction Metric Calculation

Analyzed *interactions* are RFSs and responses sent by an actor n_i regarding another one n_j. The *context* of interactions reflects the situation and reason for their occurrences, and is modeled as activities. When interactions are interpreted, only a minor subset of all describing context elements is relevant within a *trust scope*. In the motivating scenario, such a trust scope may describe the expertise area that is required to process an RFS.

```xml
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:vietypes="http://viete.infosys.tuwien.ac.at/Type"
   xmlns:hps="http://hps.infosys.tuwien.ac.at/"
   xmlns:rfs="http://viete.infosys.tuwien.ac.at/Type/RFS">
   <soap:Header>
      <vietypes:timestamp value="2009-03-05T15:13:21"/>
      <vietypes:delegation hops="3" deadline="2009-03-06T12:00:00"/>
      <vietypes:activity url="http://www.coin-ip.eu/Activity#42"/>
      <wsa:MessageID>uuid:722B1240-...</wsa:MessageID>
      <wsa:ReplyTo>http://viete.infosys.tuwien.ac.at/Actor#Florian</wsa:ReplyTo>
      <wsa:From>http://viete.infosys.tuwien.ac.at/Actor#Florian</wsa:From>
      <wsa:To>http://viete.infosys.tuwien.ac.at/Actor#Daniel</wsa:To>
      <wsa:Action>http://viete.infosys.tuwien.ac.at/Type/RFS</wsa:Action>
   </soap:Header>
   <soap:Body>
      <hps:RFS>
         <rfs:requ>Can you create an ant file for projectX?</rfs:requ>
         <rfs:generalterms>programming</rfs:generalterms>
         <rfs:keywords>java, EE, ant, apache axis2</rfs:keywords>
         <rfs:resource url="http://svn.vitalab.tuwien.ac.at/projectX"/>
      </hps:RFS>
   </soap:Body>
</soap:Envelope>
```

Listing 4.1: Simplified RFS via SOAP example.

Table 4.3.2 shows some example interaction metrics suitable for trust interpretation that can be calculated from logged SOAP calls. Note, as described before, these metrics are determined for particular scopes. The availability of a service, either provided by humans or implemented in Software, can be high in one scope, but much lower in another one. Furthermore, these metrics are calculated for each directed relation between pairs of network members. Note, an actor n_i might serve n_j reliably, but not a third party n_k.

metric name	range	unit	description
availability	[0,100]	%	ratio of accepted to all received RFSs
response time	[0,96]	hours	average response time in hours
success rate	[0,100]	%	amount of successfully served RFSs
experience	[0,∞]	1	number of RFSs served
RFS reciprocity	[-1,1]	1	ratio of processed to sent RFSs
manual reward	[0,5]	1	optional manually assigned scores
costs	[0,5]	$	price for serving RFSs

Table 4.1: Metrics utilized for trust inference.

For the sake of brevity, in the following examples and evaluation we account only for the *average response time* t_r (Equation 4.1) of a service and its *success rate* *sr* (Equation 4.2). These are typical metrics for an *emergency help and support environment*, where fast and reliable support is absolutely required, but costs can be neglected. We assume, similar complexity of requests for support (RFS) in a scope s, thus different RFSs require comparable efforts from services (similar to a traditional Internet forum).

The response time is calculated as the duration between sending (or delegating) a request (t_{send}) to a service and receiving the corresponding response ($t_{receive}$), averaged over all served RFSs. Unique IDs of calls (see SOAP header in Listing 4.1) enable sophisticated message correlation to identify corresponding messages.

$$t_r^s = \frac{\sum_{rfs \in RFS} (t_{receive}(rfs) - t_{send}(rfs))}{|RFS|} \tag{4.1}$$

An RFS is considered successfully served ($sRFS$) if leading to a result before a predefined deadline, otherwise it fails ($fRFS$).

$$sr^s = \frac{num(sRFS)}{num(sRFS) + num(fRFS)} \tag{4.2}$$

4.3.3 Interpretation and Trust Inference

On top of the interaction metrics M of n_i towards n_j in scope s (here: $M = \{t_r^s, sr^s\}$), personal trust $\tau^s(n_i, n_j) \in [0,1]$ is inferred. Trust, describing the relationship from n_i to n_j, represents recent evidence that an actor behaves dependably, securely and reliably. The function Ψ^s (Equation 4.3) evaluates metrics M with a rule set R to interpret trust τ in scope s.

$$\tau^s(n_i, n_j) = \Psi^s(n_i, M(n_i, n_j), R, s) \tag{4.3}$$

Instead of usual business rules, we utilize a fuzzy set theory approach [113] that enables an intuitive way to combine and interpret various metrics as trust from a collaborative and social point of view. Fuzzy set theory, developed by Zadeh [131], and fuzzy logic emerged in the domain of control engineering, but are nowadays increasingly used in computer science to enable lightweight reasoning on a set of imperfect data or knowledge. The concept of fuzziness has been used earlier in trust models [45, 96, 109], however, to our best knowledge not to enable an interpretation of trust from larger and diverse sets of metrics, calculated upon observed interactions.

As fuzzy inference is a key mechanisms of our trust model, we introduce the fundamental definitions in this section. Further details are, for instance, in [136]. Zadeh [131] defined a *fuzzy set* A in X ($A \subseteq X$) to be characterized by a membership function $\mu_A(x) : X \mapsto [0,1]$ which associates with each point in X a real number in the interval $[0,1]$, with the value of $\mu_A(x)$ at x representing the 'grade of membership' of x in A. Thus, the nearer the value of $\mu_A(x)$ to 1, the higher the grade of membership of x in A. When A is a set in the ordinary sense of the term, its membership function can take only two values ($\mu_A(x) : X \mapsto \{0,1\}$, Equation 4.4), according as x does or does not belong to A. Thus, in this case $\mu_A(x)$ reduces to the familiar characteristic function of a set A.

$$\mu_A(x) = \begin{cases} 1 & \text{if x} \in \text{A} \\ 0 & \text{if x} \notin \text{A} \end{cases} \tag{4.4}$$

Equation 4.5 depicts an example definition of a membership function $\mu_A(x)$ describing a fuzzy set. This membership function is part of the linguistic variable 'responsiveness' highlighted in Figure 4.3(a), left side.

$$\mu_A(x) = \begin{cases} 0 & \text{if } 0 \leq x < 12 \\ \frac{x}{12} - 1 & \text{if } 12 \leq x < 24 \\ 1 & \text{if } 24 \leq x < 48 \\ -\frac{x}{12} - 5 & \text{if } 48 \leq x < 60 \\ 0 & \text{else} \end{cases} \tag{4.5}$$

Two or more fuzzy sets, describing the same characteristic (i.e., metric), can be merged to a *linguistic variable*. For instance in Figure 4.3(a), the linguistic variable 'responsiveness' is described by three fuzzy sets: high, medium, and low.

The definition of linguistic variables (and the their single membership functions respectively), has to be performed carefully as they determine the operation of the reasoning process. Linguistic variables are defined either for the whole community, or for groups, and even single network members, by:

- a domain expert, using his experience and expertise. However, depending on the complexity of the rules and aggregated metrics continuous manual adjustments are needed (especially when bootstrapping the trust system).
- the system itself based on knowledge about the whole community. For instance, the definition of a 'high' skill level is determined by the best 10 percent of all network members in certain areas.
- the users based on individual constraints. For example, a 'high' skill level from user n_i's point of view starts with having more than the double score of himself.

Let X_A and X_B be two feature spaces, and sets that are describes by their membership function μ_A and μ_B respectively. A *fuzzy relation* $\mu_R(x_A, x_B) : X_A \times X_B \mapsto [0,1]$ describes the set X, whereas $\mu_R(x_A, x_B)$ associates each element (x_A, x_B) from the cartesian product $X_A \times X_B$ a membership degree in $[0,1]$. Fuzzy relations are defined by a rule base, where each rule, as shown in Equation 4.6, comprises a premise p (condition to be met) and a conclusion c.

$$\textbf{IF } p \textbf{ THEN } c \tag{4.6}$$

Approximate reasoning by evaluating the aforementioned rule base, needs some fuzzy operators to be defined [131]: **OR**, **AND**, and **NOT**.

$$A \textbf{ OR } B \equiv A \cup B \equiv \mu(x) = max(\mu_A(x), \mu_B(x)) \text{ for } x \in X \tag{4.7}$$

$$A \textbf{ AND } B \equiv A \cap B \equiv \mu(x) = min(\mu_A(x), \mu_B(x)) \text{ for } x \in X \tag{4.8}$$

$$\textbf{NOT } A \equiv \mu(x) = 1 - \mu_A(x) \text{ for } x \in X \tag{4.9}$$

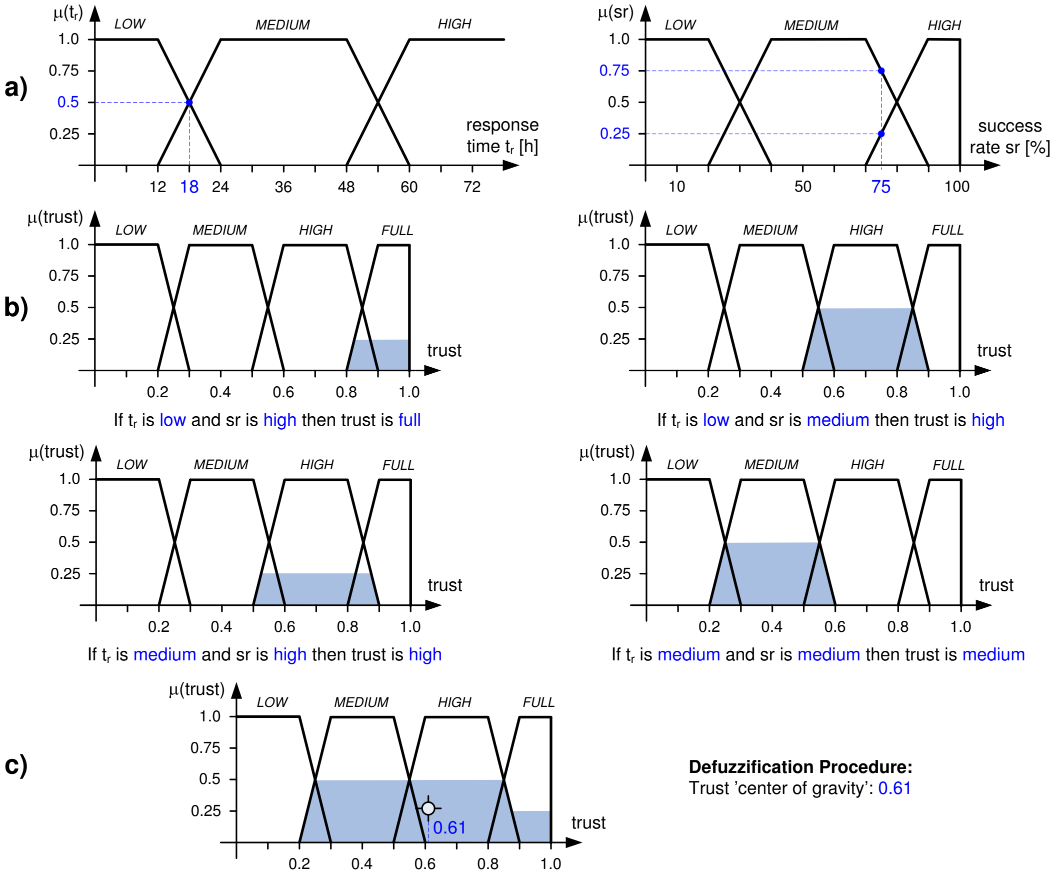


Figure 4.3: An example showing fuzzy trust inference. Applied interaction metrics are response time $t_r = 18h$ and success rate $sr = 75\%$. (a) definition of membership functions and fuzzified interaction metrics; (b) four applied fuzzy rules following the max-min inference; (c) defuzzification by determining the center of gravity.

The *defuzzification* operation [67] determines a discrete (sharp) value x_s from the inferred fuzzy set X. For that purpose all single results obtained by evaluating rules (see Figure 4.3(b)) are combined, forming a geometric shape. One of the most common defuzzification methods is to determine the *center of gravity* of this shape, as depicted in the example in Figure 4.3(c). In general, center of gravity defuzzification determines the component x of x_s of the area below the membership function $\mu_x(x)$ (see Equation 4.10).

$$x_s = \frac{\int_x x \cdot \mu_x(x) \cdot dx}{\int_x \mu_x(x) \cdot dx} \tag{4.10}$$

Example: Given the linguistic variables *response time* t_r, *success rate sr*, and *trust*, with the membership functions as defined in Figure 4.3, we provide the rulebase in Listing 4.2 to the fuzzy engine. Figure 4.4 visualizes trust inference results for different pairs of t_r and sr inputs.

```
if $t_r$ is low    and $sr$ is high   then $trust$ is full
if $t_r$ is low    and $sr$ is medium then $trust$ is high
if $t_r$ is low    and $sr$ is low    then $trust$ is low
if $t_r$ is medium and $sr$ is high   then $trust$ is high
if $t_r$ is medium and $sr$ is medium then $trust$ is medium
if $t_r$ is medium and $sr$ is low    then $trust$ is low
if $t_r$ is high   and $sr$ is high   then $trust$ is medium
if $t_r$ is high   and $sr$ is medium then $trust$ is low
if $t_r$ is high   and $sr$ is low    then $trust$ is low
```

Listing 4.2: Rules for inferring trust upon t_r and sr.

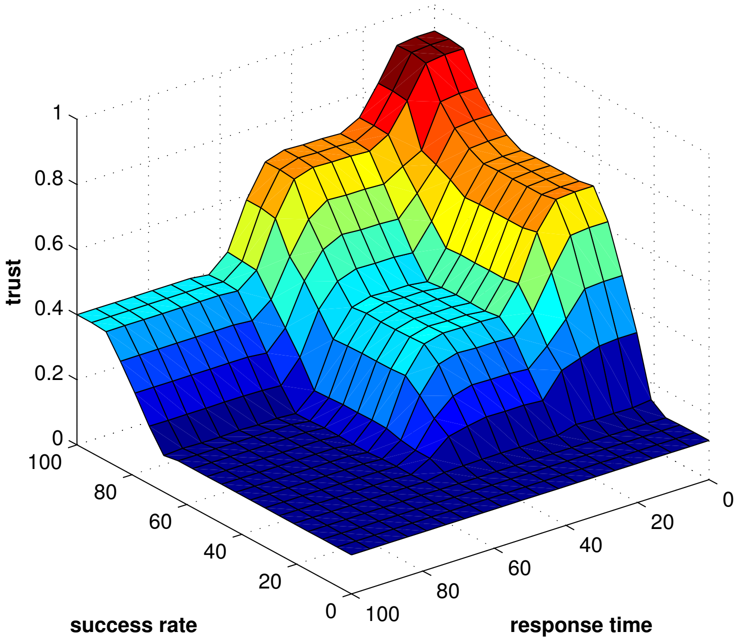


Figure 4.4: Result space for the given rule set.

Personal trust $\tau^s(n_i, n_j)$ from n_i in n_j is updated periodically in consecutive time intervals (e.g., on a daily basis). We apply a sliding window approach and process all logged interactions within a pre-configured time frame (e.g., the last week or month). The size of the window depends on the calculated interaction metrics. For instance, success rates or collected experiences are inferred from interactions within the last six month, while response times only depend on interactions in the last week. Therefore, the depth of the required interaction history relies on the utilized metrics of the environment. We demonstrate an application of this approach in the evaluation in Section 4.5.

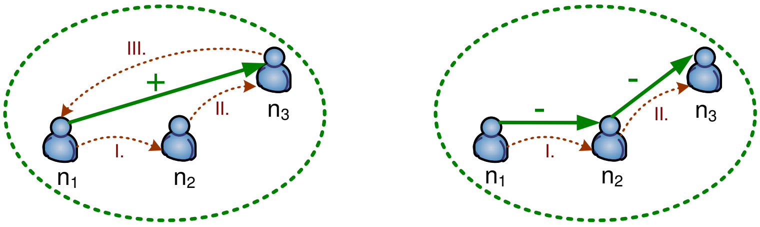
4.3.4 Collaboration Network Provisioning

Finally, the social network, comprising actors connected by trust relations, is provided by VieTE through a SOAP interface (see top of Figure 4.2). A trust relation is always asymmetric, i.e., a directed edge from one member node to another one in a graph model $G = (N, E)$. We call the trusting actor the *trustor* n_i (the source of an edge), and the trusted entitiy the *trustee* n_j (the sink of an edge). VieTE's provisioning interface, described by WSDL, supports convenient network retrieval operations, such as getting the trustors and trustees of a node in a specified scope (see Appendix).

A domain expert configures certain properties of the trust inference process that are applied for all participants of the network. For instance, s/he defines meaningful trust scopes in the given domain and business area, configures available metric calculation plugins that provide the metrics for personal trust rules, and sets up the general trust model behavior, such as temporal constraints for interaction analysis and endpoints of logging facilities.

4.4 Delegations and Balancing

A common problem of trust and reputation mechanisms in online communities is that there emerge only a minority of highly trusted actors, while the majority remains in the background. Therefore, network members tend to consult and interact with the same (already trusted) services again and again, leading to work overloads of these service providers, and hindering the emergence of new trust relations. We utilize the means of delegations to compensate this load and interaction balancing problem that is often neglected, but of paramount importance in collaborative environments. In the motivating PVC scenario of this chapter, actors send and process requests for support (RFS). Once an actor gets overloaded s/he should be able to delegate requests to other actors (with potentially free resources). If the receivers of such delegations behave trustworthy, i.e., respond fast and reliably, the original requesters will establish trust to them. Figure 4.5 visualizes this model. In case of a successful delegation, n_1 sends an RFS to n_2 who delegates to n_3, and n_3 responds directly to n_1. This interaction will positively impact the metrics that describe the relation from n_1 to n_2, and finally $\tau(n_1, n_2)$ increases. The relation $\tau(n_1, n_2)$ is neither rewarded nor punished, because on the one side n_2 did not serve n_1's RFS, but on the other side, n_2 was able to successfully delegate, and thus did not harm n_1. The relation $\tau(n_2, n_3)$ is also not influenced, since n_2 is not the original requester. If a delegation fails (Figure 4.5(b)), i.e., an RFS is not responded, metrics that describe both $\tau(n_1, n_2)$ and $\tau(n_2, n_3)$ are negatively influenced (for instance the success rate is decreased), because of n_2's and n_3's unreliable behavior. But in that case, we assume that $\tau(n_1, n_3)$ remains unchanged. Although n_3 has not served n_1's request, we do not know the reasons for that behavior. For instance, a denial of service attack could maliciously harm n_3's reputation (the average of trust relations), if s/he is flooded with delegated RFSs.



(a) Successful delegation. (b) Failed delegation.

Figure 4.5: Delegations and their impact on trust.

The described delegation mechanisms and their influence on trust are configured by a domain expert in VieTE, and are feasible for our mixed service systems PVC scenario, where all participants in the network have similar collaboration roles (in particular to provide help and support). Other delegation and trust mechanisms, accounting for different roles of network members and restrictions of delegations due to confidentiality reasons, may be desirable in other domains.

One of the major challenges to enable sophisticated balancing is to determine the receivers of delegations in the whole network. Usually, the selection will rely on trust, because, as shown in Figure 4.5(b), it is in n_2's interest to delegate successfully and not to get punished. A fundamental selection strategy randomly picks an actor from a pool of service providers that are personally trusted above a pre-defined limit. Based on each individual's interaction history, every network member has his/her own pool of trusted actors. More advanced selection models are out of scope of this work, and are subject to further research.

4.5 Evaluation and Discussion of a PVC Scenario

We evaluate the VieTE framework that implements our approach of fuzzy trust inference and balancing, by simulating typical scenarios in the described PVC environment. For that purpose, we utilize the popular Repast Simphony[3] toolkit, a software bundle that enables round-based agent simulation. In contrast to researchers in the agent domain, we do not simulate our concepts by implementing different actor types and their behavior only, but we use a network of actors to provide stimuli for the actual VieTE framework. Therefore, we are not only able to evaluate our new approach of fuzzy trust inference, but also the technical grounding based on Web service standards. Figure 4.6 depicts that VieTE is used exactly in the same manner in our simulation (highlighted left side), as it would be used by a real mixed systems PVC (right side).

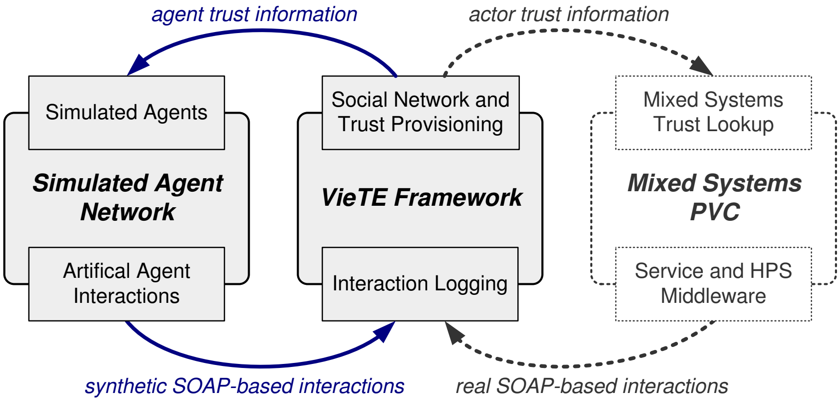

agent trust information
actor trust information
Simulated Agents
Simulated Agent Network
Artifical Agent Interactions
Social Network and Trust Provisioning
VieTE Framework
Interaction Logging
Mixed Systems Trust Lookup
Mixed Systems PVC
Service and HPS Middleware
synthetic SOAP-based interactions
real SOAP-based interactions

Figure 4.6: Simulation setup (left side) compared to the intended application (right side) of the VieTE framework.

[3] http://repast.sourceforge.net

In particular, we let the simulated network members interact (sending, responding, and delegating RFSs), and provide these interactions to the logging facilities of VieTE. The framework infers trust by calculating the described metrics t_r and sr, and using the rule set of Listing 4.2 for behavioral interpretation. Finally, emerging trust relations between the simulated actors influence the selection of receivers of RFSs. Hence, VieTE and the simulated actor network relies on each other, and are used in a cyclic approach; exactly the same way VieTE would be used by real PVCs. To facilitate simulations, all interactions take place in the same scope.

4.5.1 Simulation Setup

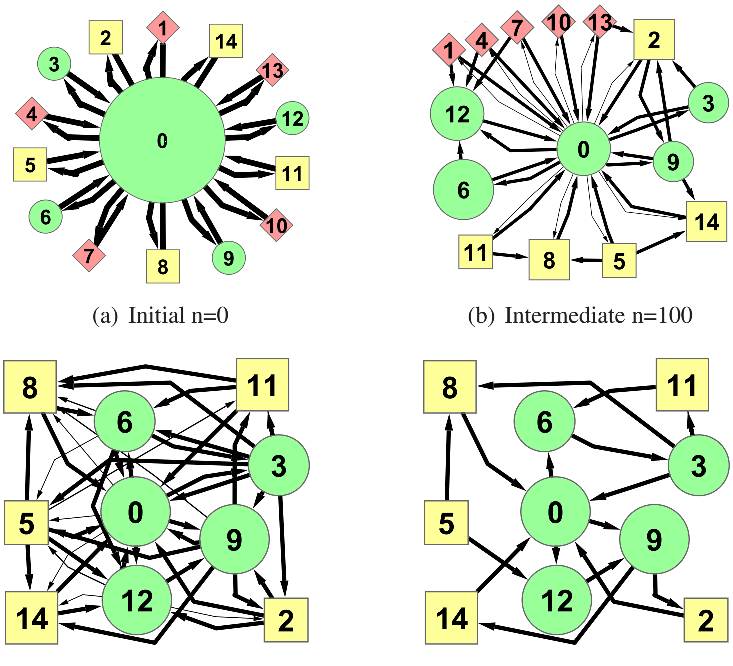
Simulated Agent Network. Repast Simphony offers convenient support to model different actor behavior. As an inherent part of our environment, we make no distinction between human users and software services. Each actor owns a unique id (a number), produces SOAP requests, and follows one of the following behavior models: (i) *malicious actors* accept all RFSs but never delegate or respond, (ii) *erratic actors* accept all RFSs but only process (respond directly or delegate) RFSs originally coming from requesters with odd-numbered IDs, (iii) *fair players* process all requests if they are not overloaded, and delegate to trustworthy network neighbors otherwise.

We set up a network comprising 15 actors, where only one is highly reputed and fully trusted by all others as depicted in Figure 4.7(a). This is the typical starting point of a newly created community, where one actor invites others to join.

VieTE Setup. After each simulation step (round) seven randomly picked actors send one RFS to its most trusted actor (in the beginning this will only be the highly reputed one who starts to delegate). Each actor's input queue has exactly 5 slots to buffer incoming RFSs. A request is always accepted and takes exactly one round to be served. An actor processes an RFS itself if it has a free slot in its input queue, otherwise incoming RFSs are delegated to randomly picked trusted ($\tau > 0.8$) neighbors in the network. Note, one actor does not delegate more than one RFS per round to the same neighbor, however, an actor may receive more than one RFS from different neighbors in the same round. Delegations require one additional simulation round. There is an upper limit of 15 rounds for an RFS to be served (deadline); otherwise it is considered failed. A request can be delegated only three times (but not back to the original requester) ($hops$) to avoid circulating RFSs. Because the simulation utilizes only two fully automatically determined metrics (t_r and sr), and no manual rewarding of responses, we assume an RFS is successfully served if a response arrives within 15 rounds (no fake or low quality responses). After each round, VieTE determines t_r based on interactions in the last 25 rounds, and sr upon interactions in the last 50 rounds (sliding window approach), and purges older logs.

4.5.2 Simulation Results

Interaction Balancing. We perform 250 simulation rounds of the described scenario with the aforementioned properties, and study the network structure in certain points of the simulation. The depicted networks in Figure 4.7 show actors with different behavior and



(a) Initial n=0

(b) Intermediate n=100

(c) Balanced n=250

(d) Balanced (reduced)

Figure 4.7: Network structure after simulation round n={0, 100, 250}. Elliptic nodes are fair players, rectangular shapes represent erratic actors, diamond shaped nodes reflect nodes with malicious behavior.

the temporal evolvement of trust relations between them. The size of the graph's nodes depend on the amount of trust established by network neighbors. Beginning with a star structure (Figure 4.7(a)), the network structure in Figure 4.7(b) emerges after 100 rounds, and Figure 4.7(c) after 250 rounds respectively. Note, since the behavior of the nodes is not deterministic (i.e., RFSs are sent to random neighbors that are trusted with $\tau > 0.8$ (lower bound of *full trust*; see Figure 4.3)), the simulation output looks differently for each simulation run, however, the overall properties of the network are similar (number and strength of emerged trust relations).

In the beginning, all RFSs are sent to *actor 0*, who delegates to randomly picked trusted actors. If they respond reliably, then requesters establish trust in that third parties. Otherwise they lose trust in *actor 0* (because of unsuccessful delegations). Therefore, actors with even-numbered IDs lose trust in *actor 0* faster than odd-numbered actors, because if *actor 0* delegates requests to erratic actors, they are not replied. As an additional feature in round 100, actors that are not trusted with $\tau > 0.2$ by at least on other network member, are removed from the network, similar to Web communities where *leechers* (actors that do not contribute to the network) are banned. Therefore, actors with malicious behavior disappear, while actors with erratic behavior still remain in the network. Figure 4.7(d) shows a reduced view of the balanced network after 250 rounds. Only trust relations with $\tau > 0.8$ are visualized. As expected most nodes have strong trust relations in at least one fair player (actors who reliably respond and delegate RFSs). However, remember that erratic actors

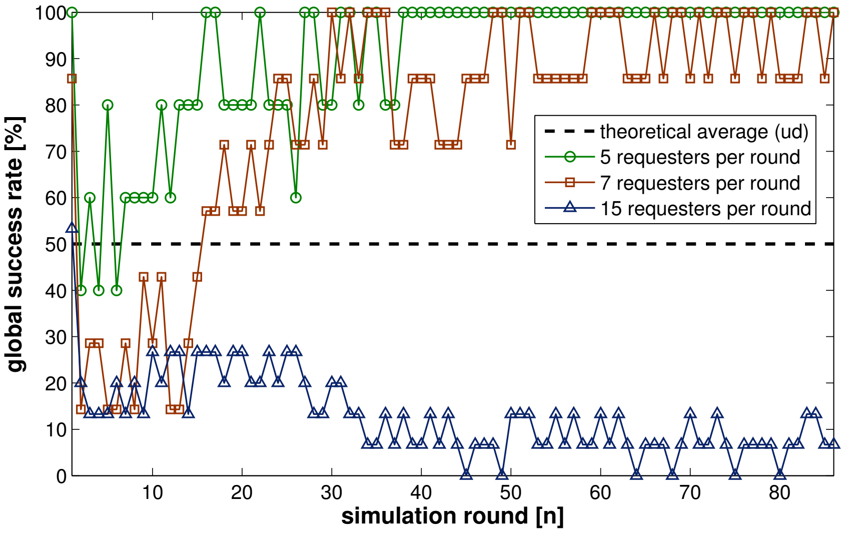


Figure 4.8: Global RFS response success rate of the simulated actor network.

reliably serve only requests coming from actors with odd-numbered IDs. Therefore, *actor 3* and *actor 9* also establish full trust in actors from this class. Note, if *actor 3* and *actor 9* would have re-delegated much RFSs coming from even-numbered actors to erratic actors, than those RFSs would have failed and only low trust would have emerged. However, due to the comparatively low load of the network (less than half of the actors receive RFSs per round (until $n = 100$)), only a low amount of re-delegations occur (approx. 8 percent).

Global Success Rate. We run the simulation with different interaction rates of actors. In particular, we let 5, 7 (as in the experiment described before), and 15 actors send RFSs to others, and calculate the global success rate, i.e., the amount of successfully answered RFSs from a global point of view. If requests would not be sent to trusted actors that proved their reliable behavior before, but uniformly distributed over available actor classes (5 fair, 5 erratic, 5 malicious) – according to a primitive interaction balancing approach – than 50 percent of RFSs would be served successfully. This theoretical limit (without delegations) is represented as a reference in Figure 4.8 by the dashed line. We study the performance of our trustworthy interaction balancing approach compared to this primitive method.

In Figure 4.8 some results are visualized, and the following issues are worth mentioning. The deadline for an RFS to be served is 15 rounds. So, the success rate of round i is available in round $i + 15$. We simulate the actor behavior until round 100 (where malicious actors are banned). Therefore, the temporal evolution of the global success rate is depicted until round 85. In the beginning, i.e., the very first round, the success rate is very high, because all RFSs are sent to *actor 0* (a fair player), who performs/buffers the first five RFSs itself (and then begins to delegate). Hence, the success rate is high for the first round, but collapses in the second round, where virtually only delegations occur. The rate slowly recovers as new trust relations emerge, and members get connected to new potential actors to send RFSs to.

In the simulation, where each of five randomly picked actors send one request to another one, the success rate stabilizes at a high level, after new trust relations emerged, and the network has been balanced. This process can be studied for seven actors again. However, since it comes to re-delegations, some RFSs frequently fail being processed (due to the impact of erratic actors), and therefore, the success rate oscillates at a high level. The case of 15 interacting network members per round shows, that actors are mainly busy with re-delegations and the large part of RFSs miss their deadlines. This results in a much lower success rate than the theoretical average of 50%.

Finally, note that our interaction balancing model of trustworthy delegations performs best, if the overall amount of reliable actors is high, and the load is low to medium.

CHAPTER 5

Trusted Interaction Patterns in Enterprise Service Networks

Outline. We refine our delegation concepts, and discuss the interplay of interaction patterns and trust. For that purpose, we formulate our trust model and show its application in context of the Expert Web use case.

Contents

5.1 Motivation

Collaborations on the Web and in large-scale enterprises evolve in a rapid pace by allowing people to form communities and expertise clusters depending on their skills and interests. The management of interactions in these networks becomes increasingly complex as current tools only support messaging and addressing mechanisms developed for the early Web.

However, it is difficult - if not impossible - to control interactions ranging from ad-hoc to process-centric collaborations. In Web-scale networks, partially unknown participants might be part of processes that span multiple geographically distributed units of an organization. We argue that trusted selection of participants leads to more efficient cooperation and compositions of human- and software services. In this chapter, we describe our novel trust concepts, including trust inference and reputation management, in the context of an enterprise use case. Trust is built upon previous interactions and evolves over time; thus providing a reliable and intuitive way to support actor selection in mixed systems.

Our key contributions in this chapter are as follows:

- *Formalized Social Trust Model.* Based on previous work, we formulate a social trust model for mixed service-oriented systems, including personal trust, recommendation, and reputation relying on measured interaction metrics. Furthermore, we deal with basic temporal evaluations.
- *Interaction Patterns.* We utilize our trust model for trusted expert ranking and selection in service-oriented communities. On the one hand we demonstrate how trust influences interactions in mixed systems; on the other hand we show the impact of interactions on trust.
- *Delegation Scenarios.* Coordination and compositions are strongly influenced by interaction patterns. We propose models for the propagation of trust based on delegations resulting in different types of patterns. We consider (i) trust *referral* in triad interaction patterns, and (ii) trust *attenuation* in proxy patterns.

These contributions, including the interplay of interaction patterns and trust, are described and motivated in the context of an enterprise use case. The main focus of this chapter is to present above mentioned interactions patterns, an approach for trust propagation within such patterns, and the discussion of the *TrueExpert* framework and its implementation (relying on VieTE's mechanisms).

5.2 Coordination and Composition

A motivating use case for our work – referred to as the *Expert Web use case* – is depicted in Figure 5.1. A process composed of single tasks assigned to humans or covered by software services, describes the steps to produce a Computer Aided Design (CAD) drawing, and handle its final delivery to a customer. A conceptual draft, e.g., of a mechanical part, is designed by an engineer from the development department. In parallel a CAD assistant, belonging to another organizational unit, defines common symbols for the final drawing which conform to customer's requirements and international standards. After completing these tasks, a skilled drawer produces the drawing. In the last step an assistant cares for printout and delivery to the customer, and a software service converts the drawing and stores it in an archive.

We assume, that the single task owners in this process exchange only electronic files, and interact using communication technologies. While various languages and techniques

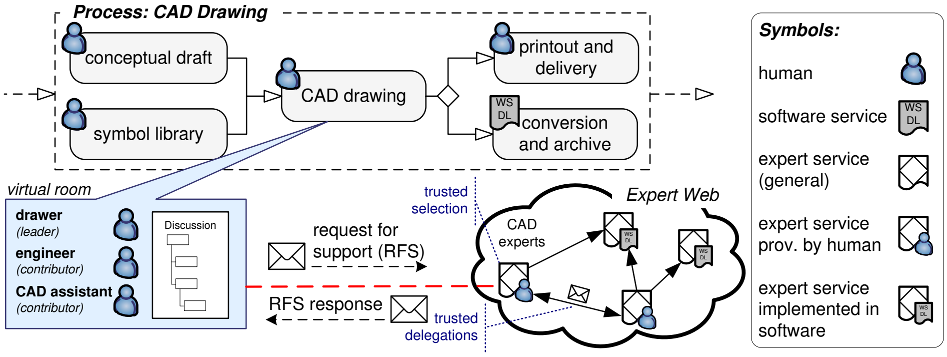


Figure 5.1: Involving experts from the Expert Web.

for modeling such processes already exist, including the Business Process Execution Language (BPEL [92]), we focus on another aspect in this scenario: *trusted online help and support*. Usually, in BPEL input and output data are rigidly specified, however, even for carefully planned processes with human participation, ad-hoc adaptation and intervention is required due to the complexity of human tasks, people's individual understanding, and unpredictable events. For instance, according to Figure 5.1 the drawer receives a drawing draft and a symbol library. If people have not yet worked jointly on similar tasks, it is likely, that they need to set up a meeting for discussing produced artifacts. Especially, if people belong to different, possibly geographically distributed organizational units, a personal meeting can be time- and cost intensive. Therefore, various Web 2.0 technologies, including forums, wiki pages and text chats, provide well-proven support for tele-work in collaborative environments (represented by the *virtual room* in Figure 5.1).

However, several challenges remain unsolved. If people, participating in the whole process, are not able to solve problems by discussion, who should be asked for support? How can third parties be contacted and informed about the current situation? How can they easily be involved in ongoing collaborations? Moreover, what are influencing factors for favoring one party over others? How is information exchanged, and how can this situation be supported by service-oriented systems?

The traditional way of discovering support is simply to ask third persons in someone's working environment, the discussion participants are convinced they are able to help, namely *trusted experts*. In an environment with a limited number of people, persons usually tend to know who can be trusted and what data has to be shared in order to proceed with solving problems of particular nature. Furthermore, they easily find ways to contact trusted experts, e.g., discovery of phone numbers or e-mail addresses. In case requesters do not know skilled persons, they may ask friends or colleagues, who faced similar problems before, to recommend experts. The drawbacks of this traditional way are that people need extensive knowledge about the skills of colleagues and internal structures of the organization (e.g., the expertises of people in other departments). The traditional way of discovering support is inefficient in large-scale enterprises with thousands of employees and probably not satisfying if an inquiry for an expert becomes a major undertaking. Even the use of today's computer-supported communication technologies cannot fully address the mentioned challenges.

The Expert Web. We propose the Expert Web, consisting of connected experts that provide help and support in a service-oriented manner. The members of this Expert Web are either *humans*, such as company employees offering help as online support services, or *software services*, encapsulating howtos[1], knowledge bases, and oracles[2] with intelligent reasoning capabilities. Such an enterprise service network, spanning various organizational units, can be consulted for efficient discovery of available support. Users, such as the engineer or drawer in our use case, send ***requests for support*** (RFSs). The users establish trust in experts' capabilities based on their response behavior (e.g., availability, response time, quality of support). This trust, reflecting personal positive or negative experiences, fundamentally influences future selections of experts.

As in the traditional case, experts may also delegate RFSs to other experts in the network, for example when they are overloaded or not able to provide satisfying responses. Following this way, not only users of the enterprise service network establish trust in experts, but also trust relations between experts emerge.

5.3 Formalized Social Trust Model

In virtual communities, where people dynamically interact to perform activities, reliable and dependable behavior promotes the emergence of trust. As collaborations are increasingly performed online, supported by service-oriented technologies, such as communication-, coordination-, and resource management services, interactions have become observable. We argue that by monitoring and analyzing interactions, trust can be automatically inferred. In contrast to manual rating approaches, automatic inference is well-suited for complex networks, where potentially thousands of network members dynamically interact. We demonstrated the automatic inference of trust earlier (see Chapter 4 and focus on a formalized model now.

5.3.1 Personal Trust Inference

Not only service interactions, but also human interactions may rely on SOAP (e.g., see Human-Provided Services [106] and BPEL4People [1]), which is the state-of-the-art technology in service-oriented environments, and well supported by various software frameworks. This fact enables the adoption of various available monitoring and logging tools for mixed service-oriented systems. The XML-based structure of SOAP messages is well-suited for message header extensions, such as addressing and routing information, and annotation with contextual elements (e.g., activity identifier). These mechanisms allow for context-aware interaction metric calculation, for instance, reliability, responsiveness, collected experience, and costs with respect to specific situations. We apply an arithmetic calculation of trust based on these metrics. This calculation is context dependent, so in different domains and use cases the impact of metrics varies. As interaction behavior changes

[1] `http://www.ehow.com`

[2] `http://www.wolframalpha.com`

over time, trust will alter too. Therefore, trust deems to be an intuitive grounding for flexible adaptation techniques (for instance, service replacement) in mixed service-oriented systems.

The interaction behavior of a network member n_i toward n_j is described by various metrics $M(n_i, n_j)$, such as average response time, availability, and rate of joint successful activities. In this work, we basically deal with two different approaches to infer trust upon these metrics:

- *Arithmetic Calculation.* This means, trust is calculated by weighting metrics and building average values. However this method uses very simple models, they may be quite effective under certain assumptions (e.g., simple interaction types and patterns).
- *Rule-based Interpretation.* This allows to account for more complex business rules to actually interpret metrics. For instance, actors need to reach certain scores of predefined metrics to be considered trustworthy.

Arithmetic Calculation. Interaction metrics are normalized to the interval $[0, 1]$ either according to predefined upper and lower bounds, or dynamically adapted according to the highest and lowest values in the whole community. Furthermore, weights need to be specified, either by users or system administrators. The weighted sum of selected metrics build the **confidence** $c^s(n_i, n_j) \in [0, 1]$ of n_i in n_j in scope s (e.g., a certain expertise area or particular type of activity). This confidence represents recent evidence that an actor behaves dependably, securely and reliably. In Equation 5.1 confidence is calculated from metrics $m_k \in M(n_i, n_j)$ that are weighted by w_k ($\sum_k w_k = 1$).

$$c^s(n_i, n_j) = \sum_{\forall m_k} (m_k(n_i, n_j) \cdot w_k) \tag{5.1}$$

Rule-based Interpretation. Interaction metrics are processed by individually configured fuzzy (E)CA[3]-rules. These rules define conditions to be met by metrics M for interpreting trustworthy behavior, e.g., 'the responsiveness of the trustee must be *high*' or 'a trustworthy software programmer must have collected at least *average* experiences in software integration activities'. Rules reflect a user's trust perception, e.g., pessimists may demand for stricter trustworthy behavior, than optimists. So, these rules are again either specified by the users, or globally by administrators. Node, only the individual definition of rules allows the expression of personalized *trust requirements*. However, individual rules highly increase the computational complexity for trust inference.

On top, again the **confidence** $c^s(n_i, n_j) \in [0, 1]$ of n_i in n_j in scope s is determined. This confidence may rely on a wide variety of interaction-, collaboration-, and similarity metrics M that describe the relationship from n_i to n_j. Besides highly dynamic interaction metrics, information from actor profiles P may be considered during calculation, e.g., a human actor's education or a service's vendor. The function Ψ_c^s (Equation 5.2) evaluates n_1's personal fuzzy rule set $R_c(n_i)$ (or a general rule set R_c) to determine confidence c in scope s in his collaboration partners (e.g., n_j) (see Chapter 4 for more information on the

[3] (event)-condition-action

application of fuzzy set theory for trust inference). This confidence value is normalized to $[0,1]$ according to the degree of rule satisfaction.

$$c^s(n_i, n_j) = \Psi_c^s(n_i, M(n_i, n_j), P(n_j), R_c(n_j), s) \tag{5.2}$$

The **reliability of confidence** $\rho(c^s(n_i, n_j)) \in [0,1]$, ranging from totally uncertain to fully confirmed, depends mainly on the amount of data used to calculate confidence (more data provide higher evidence), and the variance of metric values collected over time (e.g., stable interaction behavior is more trustworthy). The function Ψ_ρ^s (Equation 5.3) determines the reliability ρ of the confidence value $c^s(n_i, n_j)$ relying on utilized metrics $M(n_i, n_j)$. The specific implementation is out of scope of this paper.

$$\rho(c^s(n_i, n_j)) = \Psi_\rho^s(n_i, M(n_i, n_j), s) \tag{5.3}$$

We infer **personal trust** $\tau^s(n_i, n_j) \in [0,1]$ by combining confidence with its reliability (see operator $\otimes$ in Equation 5.4). This can be performed either rule-based by attenuating confidence respecting reliability, or arithmetically, for instance by multiplying confidence with reliability (as both are scaled to the interval $[0,1]$). Since trust relies directly on confidence that may be inferred by evaluating personal rules, an actor's personal trust relation in this model indeed reflects its subjective criteria for trusting another actor. Trust is managed in a directed graph $G = (N, E)$.

$$\tau^s(n_i, n_j) = \langle c^s(n_i, n_j), \rho(c^s(n_i, n_j)), \otimes \rangle \tag{5.4}$$

We introduce the **trust vector** $\mathbf{T}^s(n_i)$ to enable efficient trust management in the *Web of Trust*. This vector is combined of single personal trust relations (outgoing edges of a vertex in $G = (N, E)$) from an actor n_i to others in scope s (Equation 5.5).

$$\mathbf{T}^s(n_i) = \langle \tau^s(n_i, n_j), \tau^s(n_i, n_k), \tau^s(n_i, n_l), \dots \rangle \tag{5.5}$$

The **trust matrix** $\mathfrak{T}^s$ comprises trust vectors of all actors in the environment, and is therefore the adjacency matrix of the mentioned trust graph $G = (N, E)$. In this matrix, as shown in Equation 5.6 for four vertices $N = \{n_1, n_2, n_3, n_4\}$, each row vector describes the trusting behavior of a particular actor ($\mathbf{T}^s$), while the column vectors describe how much an actor is trusted by others. If no relation exists, such as self-connections, this is denoted by the symbol $\perp$.

$$\mathfrak{T}^s = \begin{pmatrix} \perp & \tau^s(n_1, n_2) & \tau^s(n_1, n_3) & \tau^s(n_1, n_4) \\ \tau^s(n_2, n_1) & \perp & \tau^s(n_2, n_3) & \tau^s(n_2, n_4) \\ \tau^s(n_3, n_1) & \tau^s(n_3, n_2) & \perp & \tau^s(n_3, n_4) \\ \tau^s(n_4, n_1) & \tau^s(n_4, n_2) & \tau^s(n_4, n_3) & \perp \end{pmatrix} \tag{5.6}$$

The **trust perception** $p_\tau^s(n_i)$ represents the 'trusting behavior' of n_i, i.e., its attitude to trust others in scope s. The absolute value of $p_\tau^s(n_i)$ is not of importance, but it is meaningful to compare the trust perceptions of various actors. In case of rule-based trust interpretation, this can be performed by comparing their rule bases for trust inference (Equation 5.7), e.g., if actors account for the same metrics, or if they are shaped by optimism or pessimism (and thus have lower or higher requirements on someones behavior). Therefore, more similar rules means more similar requirements for trust establishment. A typical application of trust perception is the weighting of recommendations from multiple trustees based on the similarity of the requester's and recommender's trust perceptions.

$$sim_{percep}(p^s_\tau(n_i), p^s_\tau(n_j)) = sim(R^s_c(n_i), R^s_c(n_j) \tag{5.7}$$

5.3.2 Temporal Evaluation

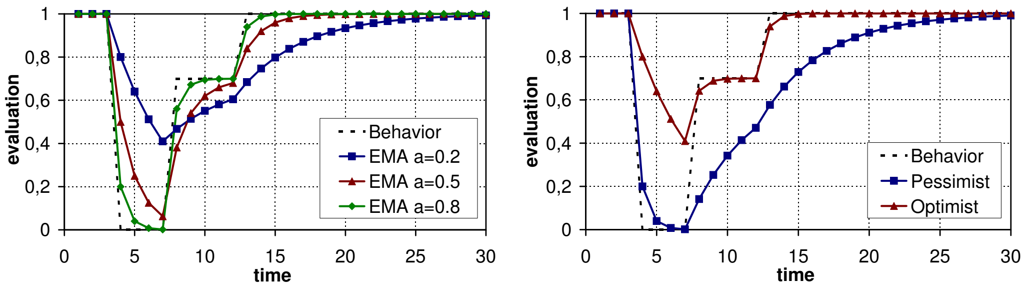
Personal trust $\tau^s(n_i, n_j)$ from n_i in n_j is updated periodically in successive time intervals t_i, numbered with consecutive integers starting with zero. We denote the personal trust value calculated at time step i as τ^s_i. As trust is evolving over time, we do not simply replace old values, i.e., τ^s_{i-1}, with newer ones, but merge them according to pre-defined rules. For this purpose we apply the concept of exponential moving average[4], to smoothen the sequence of calculated trust values as shown in Equation 5.8. With this method, we are able to adjust the importance of the most recent trust behavior $\Delta\tau^s$ compared to history trust values τ^s (smoothing factor $\alpha \in [0, 1]$). In case, there are no interactions between two entities, but an existing trust relation, the reliability of this trust relation is lowered by a small amount each evaluation interval. Therefore, equal to reality, trust between entities is reduced stepwise, if they do not interact frequently.

$$\tau^s_i = \alpha \cdot \Delta\tau^s_i + (1 - \alpha) \cdot \tau^s_{i-1} \tag{5.8}$$

Figure 5.2(a) shows an example of applied EMA. The dashed line represents the trustworthiness of an actor's behavior, i.e., $\Delta\tau^s_i$, for the i^{th} time interval, calculated independently from previous time intervals. In this situation an actor's behavior is evaluated as fully reliable, then drops to zero, and finally fully reliable again. Similar to reality, EMA enables us to memorize drops in recent behavior. If an actor once behaved untrustworthy, it will likely take some time to regain full trust again. Therefore, depending on selected α, different strategies for merging current trust values with the history can be realized. According to Equation 5.8, for $\alpha > 0.5$ the actual behavior is counted more, otherwise the history gains more importance. Figure 5.2(a) shows three smoothened time lines, calculated with different smoothing factors. There exist several other approaches to trust evolution which work with deep histories, e.g., [121], however, EMA requires less memory and lower computational effort.

As shown in Figure 5.2(a), by applying EMA previous or current behavior is given more importance. However, personal traits, such as being optimistic or pessimistic, demands for more sophisticated rules of temporal evaluation. In our case, we define an optimist as somebody who predominantly remembers positive and contributing behavior and tends to quickly forgive short-term unreliability. In contrast to that, a pessimist loses trust also for short-term unreliability and needs more time to regain trust than the optimist. Examples of this behavior are depicted by Figure 5.2(b). Optimistic and pessimistic

[4] `http://www.itl.nist.gov/div898/handbook/`



(a) Evolution of trust applying EMA.

(b) Optimistic and pessimistic perception of trust modeled with adaptive EMA.

Figure 5.2: Smoothing of trust values over time.

perceptions are realized by adapting the smoothing factor α according to Equation 5.9. Whenever the curve depicted in Figure 5.2(b) changes its sign, τ_i^s is re-calculated with adapted α. A small deviation ε denotes that the smoothing factor is either near 0 or near 1, depending on falling or rising trustworthiness. An enhanced version of this approach may adapt parameters in more fine-grained intervals, for instance, by considering lower and higher drops/rises of trustworthiness.

$$\alpha = \begin{cases} 0+\varepsilon & \text{if } \textit{optimistic} \text{ and } \tau_i^s < \tau_{i-1}^s \\ 1-\varepsilon & \text{if } \textit{optimistic} \text{ and } \tau_i^s \geq \tau_{i-1}^s \\ 0+\varepsilon & \text{if } \textit{pessimistic} \text{ and } \tau_i^s \geq \tau_{i-1}^s \\ 1-\varepsilon & \text{if } \textit{pessimistic} \text{ and } \tau_i^s < \tau_{i-1}^s \end{cases} \tag{5.9}$$

5.3.3 Trust Aggregation

Single trust relations are combined, as several times discussed before, to realize the concepts of recommendation and reputation.

Recommendation $\tau_{rec}^s(n_i, n_j)$ is built by aggregating n_i's trustees' trust relations to n_j. Recommendation represents therefore second-hand experiences. Potential recommenders of n_i for n_j are all $Rec \subseteq \{n_x \in N | \tau^s(n_i, n_x) \neq \perp \wedge \tau^s(n_x, n_j) \neq \perp\}$. The recommender's perceptions of trust will likely be different from the actor's n_i perception (that is receiving the recommendation), because all of them may define trust upon different rule sets. For instance, optimists generally tend to provide better recommendations of third parties than pessimists. Considering p_τ^s allows to account for differences in trust perceptions between the set of recommenders Rec and a user n_i of recommendations. Thus, n_i could define to utilize only recommendations of trustees having similar perceptions of trust, i.e., $p_\tau^s(n_i) \approx p_\tau^s(n_x) \forall n_x \in N$. As other models [55], we weight the recommendation of each $n_x \in Rec$ with the trustworthiness of n_i in n_x (see Equation 5.10).

$$\tau_{rec}^s(n_i, n_j) = \frac{\sum_{n_x \in Rec} \tau^s(n_x, n_j) \cdot \tau^s(n_i, n_x)}{\sum_{n_x \in Rec} \tau^s(n_i, n_x)} \tag{5.10}$$

Reputation τ_{rep} is similar to recommendation, however, an actor n_i who is determining the reputation of n_j does not require a personal trust relation to n_j's trustors ('reputing' entities $Rep_j \subseteq \{n_x \in N | \tau^s(n_x, n_j) \neq \perp\}$ described by a column vector of the trust matrix $\mathfrak{T}^s$). Reputation represents a kind of global (community) trust, calculated on top of each trustor's personal relations (see Equation 5.11). If personal trust relations rely on individually set up rules, the results of our reputation model indeed reflect someone's *real standing*, influenced by the subjective trust perceptions of his trustors. More advanced models may account for the reputation of the trustors, leading to a Page Rank-like model [93].

$$\tau^s_{rep}(n_j) = \frac{\sum_{n \in Rep_j} \tau^s(n, n_j)}{|Rep_j|} \tag{5.11}$$

5.4 Interaction Patterns and Trust

According to the motivating example in Figure 5.1, leaders of tasks in a process can invite participants of preceding tasks to a discussion in a virtual room, using Web 2.0 technologies such as discussion forums. If people are not able to find solutions for occurring problems, the discussion leader may consult expert services. These services are selected by considering the discussion leader's trust requirements (e.g., a minimum personal trust or reputation) in expert services.

5.4.1 Service Selection Procedure

We distinguish between two roles: *requesters* and *expert services*. Requesters, i.e., service users, are humans requesting support; expert services are provided by either humans (HPS) or are implemented in software. A human can be both a service requester and provider at the same time.

Algorithm 3 implements the fundamental procedure for selecting a service from the Expert Web. At first, it is determined if a scope s that sufficiently describes the current situation ($EnvDescr$) already exists (i.e., context similarity is above a predefined threshold ϑ_{Ctx}). Otherwise, a new scope is created using available data from the environment. This step is performed by the support requester who specifies context data that describes the scope, such as the type of the current problem. Afterward, a support activity is created. Then the algorithm determines if there is a sufficient number of trust relations applying to s in G_T, to reliably calculate recommendations and reputation of services. If this *penetration* of s in G_T is greater than a threshold ϑ_p, then the set of expert services N' is discovered by evaluating predefined trust requirements specified by the requester. Otherwise, if $penetration(G_T, s) < \vartheta_p$, a fallback strategy is applied and the service providers' skill profiles are compared with the problem description and requirements in s (traditional skill matching). From the pool of services that fulfill the requester's trust requirements, the system picks one based on pre-defined traditional selection rules, e.g., accounting for the vendor or QoS. Finally, the request for support (RFS) is compiled. The definitions of trust requirements and service selection rules are presented in Section 5.5.

Algorithm 3 Create RFS of user n_i to request support from service n_j

```
Require: trust network $G_T = (N, E)$, user $n_i$, $problem$ description, trust $policy$
/* determine $s$, create support activity */
$Scope[\,] \leftarrow$ getAvailableContextualScopes$(G_T)$
$EnvDescr \leftarrow$ collectContextElements$(n_i)$
$s \leftarrow$ maxSimilarity$(Scope[\,], EnvDescr)$
if (sim$(EnvDescr, s) < \vartheta_{Ctx}) \vee (s = \emptyset)$ then
    $s \leftarrow$ createProblemScope$(EnvDescr)$
    addScopeToTrustNetwork$(G_T, s)$
$p \leftarrow$ createProblemDefinition$(n_i, problem)$
$a \leftarrow$ createSupportActivity$(p, s, policy)$
/* discover list of potential services */
if penetration$(G_T, s) \geq \vartheta_p$ then
    /* discovery based on trust requirements */
    $N' \leftarrow$ evalTrustRequirements$(n_i, N, s)$
else
    /* competency coverage fallback */
    $N' \leftarrow$ evalCompetencyCoverage$(N, a)$
$n_j \leftarrow$ selectService$(N', rules(n_i))$
$RFS \leftarrow$ createRequest$(n_i, n_j, a)$
return $RFS$
```

5.4.2 Fundamental Interactions

The fundamental interactions when requesting support are depicted in Figure 5.3. First, the requester sends an RFS, specifying the problem and policies for treating the request. Second, the expert service sends an *intermediate answer*, informing the requester that the RFS has been accepted, rejected or delegated to other members in the Expert Web. Third, after fully processing an accepted RFS (or treating a delegated one – see later), the *final answer*, including a solution for the requester's problem, is delivered. All performed interactions are monitored and evaluated by the system. The supporting expert service gets rewarded depending on the success and outcome of interactions, and a trust relation from the requester to the service is established.

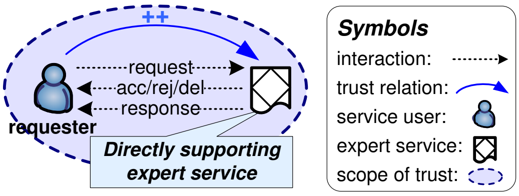


Figure 5.3: From interactions to trust.

5.4.3 Rewarding and Punishment

We utilize observed interactions to infer personal trust as discussed in the last section of this chapter. Therefore, we introduce the following methods for rewarding an expert's impact on ongoing discussions (remember the virtual room in Figure 5.1): (i) *automatic:* Availability of services and response behavior (e.g., rejecting RFSs) are determined through the means of interaction mining. (ii) *semi-automatic:* In the RFS the requester can specify the importance and a hard deadline for a response. Depending on whether an answer arrives in time, trust either increases or decreases. (iii) *manual:* The discussion leader may rate a service provider's support manually (e.g., 4 of 5 stars).

Our trust model, as described before, relies on the concept of confidence. New confidence values are calculated in fixed time intervals i, e.g., on a weekly basis. Based on earned rewards, the current confidence value at time i is calculated by updating the recent value at $i-1$. For all kinds of rewarding we apply again the concept of exponential moving average (EMA) to to avoid oscillating results (see Equation 5.12). The variable rew represents the reward, given automatically by the system, or manually by the user, for support in the last time interval; rew_{max} is the maximum possible amount (if all RFSs are served reliably). EMA weights the importance of recent rewards while not discarding older ones (smoothing factor $\alpha \in [0,1]$). Because we maintain relative levels of confidence and trust ($\in [0,1]$), services can be punished for bad support by giving comparatively low rewards, i.e., $rew << rew_{max}$.

$$c_i^s = \alpha \cdot \frac{rew}{rew_{max}} + (1-\alpha) \cdot c_{i-1}^s \tag{5.12}$$

Once confidence has been calculated, our previously introduced trust model is applied to determine and update trust, recommendation, and reputation values.

5.4.4 Requirements and Policies

Usually, organizations have strict rules, e.g., introduced by international certifications, regarding interactions across organizational units. We support this requirement and allow users to specify (i) trust requirements controlling the selection of services, and (ii) policies regulating interaction behavior.

Requirements. Users of the Expert Web can influence the service selection mechanism by specifying their requirements on trust in potential expert services. For instance, one may specify to select services with high reputation only, or the level of personal trust combined with its reliability. These configured requirements are evaluated by the system when requesting support (see Algorithm 3: function evalTrustRequirements()). A common problem of reputation systems is, that users tend to select only a small amount of top-rated services. This leads to a significantly higher use of only some services, while the majority do not receive any RFSs. Therefore, we identified the need for a trade-off model, which encourages users not to set the strictest trust requirements, but requirements appropriate in a given situation. For strict trust requirements, the number of potential service providers is usually limited, and a given RFS is either (i) assigned to an expert service after waiting for free processing time, or (ii) is rejected by the system in case there are no free services fulfilling the trust requirements. Thus, if requesters need immediate support, they will have to lower their requirements (similar to real life). This fact offers newcomers and less recognized experts the opportunity to gain reputation. More complex trade-off models account for price of service calls. For instance, experts earn rewards by providing support, and may use these rewards as payment for service usage, whereas stricter trust requirements (especially, reputation of a service) take higher amounts of payment. However, such mechanisms are out of scope of this work.

Policies. For each RFS, users may specify policies, including (i) a required intermediate answer, to get informed if a support request is accepted, rejected, or delegated, (ii) temporal constraints determining hard deadlines for intermediate and final answers, and (iii) data to be included in the request and shared with the expert service, e.g., only problem description or also further context data (e.g., current project and its resources). Especially this third point is closely related to dealing with privacy issues (see more in Chapter 11).

5.4.5 RFS Delegation Patterns

Expert services need not process all RFSs directly, but may delegate them to other expert services due to various reasons. For instance, an expert may be overloaded and therefore, not be able to process an RFS in time. Moreover, expert services may shield other services from RFSs, e.g., only a team leader receives RFSs directly that s/he delegates then to team members. We describe the influence on trust emergence for two different types of delegation patterns:

- *Triad interaction pattern* leading to *trust referral*.
- *Proxy pattern* leading to *trust attenuation*.

Triad Interaction Pattern. We introduce a triad pattern in detail (Figure 5.4) realizing delegations of RFSs within the same contextual or organizational scope, e.g., between experts in the same knowledge domain (of course, one expert may be ‘located’ in several scopes). The triad pattern is well known as *triadic closure* in social networks [128]. A *triad proxy* receives an RFS and forwards it to one of its well-trusted services. In case of complex problems, an RFS can be split into sub-requests to reduce response times. Furthermore, the complete RFS can be delegated to more than one expert service to increase reliability, i.e., the chance to get a suitable response. Final responses are not handled by the triad proxy. As all participating entities in this pattern belong to the same scope, e.g., knowledge domain, the expert service(s) may respond directly to the requester. A typical use case is load balancing in teams of people with same roles.

From the requester’s point of view, the triad proxy receives reduced rewards (Symbol +) for delegating but not processing the RFS. The actually supporting expert services receive rewards from the triad proxy, because of accepting the delegated RFS. This reward is also reduced, because the originator of the RFS is not the triad proxy, and the triad proxy has limited interest in successfully processing the request (compared to one of his own RFSs). However, the requester honors the support provided by the actually supporting expert service(s) equally compared to directly supporting services (compare Figure 5.3, symbol ++).

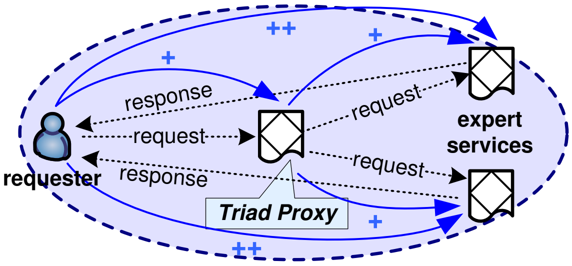


Figure 5.4: Triad interaction pattern.

Therefore, we understand highly weighted trust relations from the requester to the initial expert service, acting as a triad proxy, to be *referred* to the actual expert services.

Proxy- and Master-Slave Pattern. We adopt the well-known proxy- and master-slave patterns from the domain of software engineering [36], as applied in the domain of business interactions by [29]. In contrast to the triad pattern, the initial requester does not know the expert services an RFS is delegated to. Furthermore, the proxy may perform certain pre-processing of the RFS. The proxy pattern (forward RFS to only one expert service) and master-slave pattern (split RFS and forward to several 'slaves') are used for delegations across contextual or organizational scopes. For instance, the requester in Figure 5.5 sends an RFS to the proxy residing in the same scope. This proxy can rephrase an RFS ('translate') to be understood by an expert in another scope. The response of the expert service is processed again, before forwarding it to the initial requester. A typical example may be a head of department who acts as contact person for external people, while the actual members are not visible to the outside.

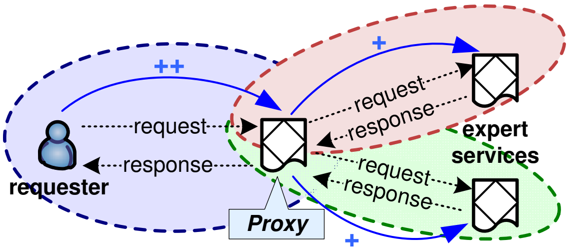


Figure 5.5: Proxy and master-slave patterns.

In contrast to the triad pattern, no trust relations from the requester to the actually supporting expert services across scopes are established. Therefore, the requester highly rewards the proxy, similar as directly supporting services. However, because the originator of the RFS is not the proxy (therefore, being less dependent on the response of the expert services), rewards given from the proxy to the expert services are reduced (equally to the behavior of the triad proxy). This leads to a trust *attenuation* from the expert service's point of view. Furthermore, the number of people or services building trust in the expert service(s) is smaller in this pattern. This reduced visibility of services' contributions has negative impact on their reputation.

5.5 TrueExpert Architecture

We depict the overview of the centralized TrueExpert architecture that supports the presented concepts, in Figure 5.6. The block on the left side contains common services from activity-centric collaboration systems (such as VieTE). On the right side, the TrueExpert services are shown. The lower layer comprises of services supporting fundamental concepts, including data access, message routing and interaction logging; on higher level services for trust management, RFS creation, and expert ranking are located. These services are utilized via SOAP- and RESTful Web service interfaces [34] from a user portal, implemented as Java Portlets[5] on top of the Liferay[6] enterprise portal.

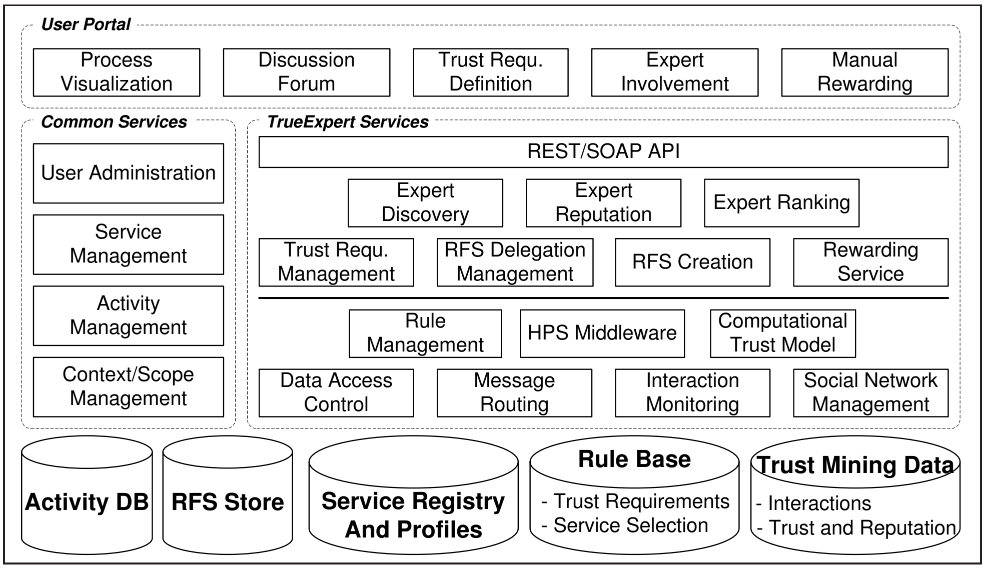


Figure 5.6: System architecture enabling trusted help and support in the Expert Web.

We outline exemplary some implementation details, focusing on the realization of the introduced trust concepts, including the RFS model, trust requirements for service discovery and selection, and delegation rules for RFS flow control.

5.5.1 Trust Requirements Rules

The trust network is implemented as directed graph using the JUNG[7] framework. In this graph model, the edges are annotated with trust metrics, i.e., personal trust, recommendation, reputation and their reliability, and references to collections of contextual information (scopes) describe situations for applying them.

We use the popular *Drools*[8] engine, to let users define their own trust requirements on potentially supporting expert services. Listing 5.1 shows an example of aggregating trust

[5] http://jcp.org/aboutJava/communityprocess/final/jsr168/

[6] http://www.liferay.com

[7] http://jung.sourceforge.net

[8] http://www.jboss.org/drools/

data about services, i.e., combine personal trust, recommendation and reputation values to myScore, which are used for ranking services in subsequent discovery operations. Before the depicted rules are applied, a user looking for support has to provide his/her own user profile and his/her current problem domain (scope), e.g., in form of activities [111]. Furthermore, copies of all available service profiles are loaded in the working memory of the rule engine. After evaluation, each service profile is temporarily personalized for the requesting user.

```
global TrustGraphDAO tgDAO;
rule "calculate score of services (without reliability)"
   salience 100
   no-loop true
      when
         service:Service()  // all potential services
         user:User()        // myself
         scope:Scope()      // my current problem scope
      then
         URI scopeId =  scope.getURI();
         URI userId = user.getUserURI();
         URI serviceId = service.getServiceURI();
         int trust = tgDAO.getPersonalTrust(userId, serviceId, scopeId);
         int rel = tgDAO.getTrustReliability(userId, serviceId, scopeId);
         int rec = tgDAO.getRecommendation(userId, serviceId, scopeId);
         int rep = tgDAO.getReputation(serviceId, scopeId);
         int score = 0.5*trust+0.3*rec+0.2*rep;    // personal weighting
         service.getPersonalization().setMetric("trust",trust);
         service.getPersonalization().setMetric("rel",rel);
         service.getPersonalization().setMetric("rep",rep);
         service.getPersonalization().setMetric("myScore",score);
end
```

Listing 5.1: Personalized trust aggregation.

Listing 5.2 shows some example rules, evaluating a user's trust requirements based on the personalized service profiles. Each service fulfilling at least one rule becomes a potentially supporting expert service. The selected-flag indicates that a service is applicable in the given scope (and for the given problem) from the user's point of view.

```
rule "Select service by average score"
   salience 50
      when
         service:Service(personalization.getMetric("myScore" >= 0.85)
      then
         service.getPersonalization().setSelected(true);
end
rule "Select personally trusted services"
   salience 50
      when
         service:Service(personalization.getMetric("trust") > 0.7 &&
                         personalization.getMetric("rel") > 0.5)
      then
         service.getPersonalization().setSelected(true);
end
```

Listing 5.2: Discover services upon requirements.

5.5.2 RFS Routing

Different system strategies for selecting one expert service from the pool of discovered services that cover a user's trust requirements, can be realized with selection rules (Listing 5.3). For instance, in case of urgent requests, the system picks services with low workload (enabling load balancing); however, for supporting risky tasks, a service with high reputation is selected.

```
rule "Urgent RFS"
   salience 50
      when
         rfs:RFS(eval(policy.flag.urgent))
         serviceList:List()  // all services fulfilling trust requirements
      then
         Service service = getServiceLWL(serviceList, 3);
         rfs.assignService(service);
end
rule "Risky RFS"
   salience 50
   when
      rfs:RFS(eval(policy.flag.risky))
      serviceList:List()  // all services fulfilling trust requirements
   then
      Service service = getServiceHR(serviceList);
      rfs.assignService(service);
end

// get services with low work load (below numRFS in queue)
function Service getServiceLWL(List serviceList, numRFS) {...}
// get service with highest reputation
function Service getServiceHR(List serviceList) {...}
```

Listing 5.3: Rules for service selection.

An excerpt of the RFS schema definitions is shown in Listings 5.4, defining complex data structures, and Listing 5.5, defining the binding of the HPS WSDL to the (HPS) infrastructure services. In the depicted example, humans offer *document review services* through HPSs. An RFS for that kind of service comprises the following main parts:

- The `GenericResource` defines common attributes and metadata associated with resources such as documents or policies. A `GenericResource` can encapsulate remote resources that are hosted by a collaboration infrastructure (e.g., document management).
- The RFS `Policy` plays an important role for controlling interaction flows, e.g., time constraints, delegation behavior including decisions whether to respond to the requester directly or to a delegating proxy, and so on.
- `Request` defines the structure of an RFS (here we show a simplified example). From the user's point of view XML Forms (XForms[9]) are used to render graphical user interfaces.
- A `Reply` is the corresponding RFS response (we omitted the actual XML defintion).

The protocol (at the technical middleware level) is asynchronous allowing RFSs to be stored, retrieved, and processed. For that purpose we implemented a middleware service (HPS Access Layer - HAL) which dispatches and routes RFSs. In Listing 5.5, `GetReview` depicts a WSDL message corresponding to the RFS `ReviewRequest`. Upon receiving such a request, HAL generates a session identifier contained in the output message `AckReviewRequest`. A notification is sent to the requester (assuming a callback destination or notification endpoint has been provided) to deliver RFS status updates for example; processed RFSs can be retrieved via `GetReviewReply`. Note, the detailed notification mechanism is not described in this work that focuses on the realization of trustworthy interaction patterns. More about HPS in detail can be found in [104].

[9] `http://www.w3.org/MarkUp/Forms/`

```
<xsd:schema tns="http://myhps.org/rfs">
   <xsd:complexType name="GenericResource">
      <xsd:sequence>
         <xsd:element name="Location" type="xsd:anyURI"/>
         <xsd:element name="Expires" type="xsd:dateTime"/>
      <xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="Request">
      <xsd:sequence>
         <xsd:element name="Policy" type="GenericResource"/>
         <xsd:element name="ReviewDoc" type="GenericResource"/>
         <xsd:element name="Comments" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="ReviewRequest" type="Request"/>
   <xsd:element name="AckReviewRequest" type="xsd:string"/>
   <xsd:element name="GetReviewReply" type="xsd:string"/>
   <xsd:element name="ReviewReply" type="Reply"/>
</xsd:schema>
```

Listing 5.4: RFS schema definition.

```
<wsdl:portType name="HPSReviewPortType">
   <wsdl:operation name="GetReview">
      <wsdl:input xmlns="http://www.w3.org/2006/05/addressing/wsdl"
          message="GetReview" wsaw:Action="urn:GetReview">
      </wsdl:input>
      <wsdl:output message="AckReviewRequest" />
   </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="HALSOAPBinding" type="HPSRFSPortType">
   <soap:binding style="document" transport="http://xmlsoap.org/soap/http"/>
</wsdl:binding>
```

Listing 5.5: WSDL RFS binding.

5.5.3 RFS Delegation

We realize delegations by the means of ECA[10] rules. While events for applying rules are hard-coded, e.g. on receiving RFSs, conditions to be met and consequences can be flexibly configured by service providers (`MyService`). Listing 5.6 shows two rules that define such a delegation policy. In this example, the RFS sender (requester) receives automatically an answer if the predefined conditions are fulfilled. Requests are (semi-)automatically delegated (if possible) when the work load is high, i.e., there are more than 10 requests waiting to be processed. Furthermore, no urgent requests are accepted on Fridays. The definition of personalized rules can be restricted for users based on contractual terms of the Expert Web.

```
rule "Delegate RFS when overloaded"
   when
      rfs:RFS()
      service:MyService(rfsQueue.length > 10)
   then
      rfs.setResponse(RFSResponse.DELEGATED, new Message("I’m really busy, so I delegated your RFS."));
end
rule "Reject urgent RFS on Fridays"
   when
      rfs:RFS(eval(flag.urgent))
      service:MyService(workingproperties.lastDayOfWeek==Calendar.FRIDAY)
   then
      if(GregorianCalendar.getInstance().get(Calendar.DAY_OF_WEEK) == Calendar.FRIDAY)
         rfs.setResponse(RFSResponse.REJECTED, new Message("It’s Friday, don’t stress me out!"));
end
```

Listing 5.6: RFS acceptance and delegation rules.

[10] Event-Condition-Action

5.5.4 Rewarding Mechanisms

Based on interaction success and reliability of services, configured rules calculate rewards. These rules evaluate interaction metrics, calculated on top of interaction logs. Our implementation accounts for the fundamental interaction metrics that have been presented in Table 4.3.2 in Chapter 4.

Figure 5.7 visualizes in a flow chart what relations get rewarded and punished respectively according to delegation behavior. A requester r sends an RFS to a proxy p, who either processes this request itself or delegates to a further service s_x. The symbol $\tau(n_i, n_j)+$ indicates that the relation from n_i to n_j gets rewarded, i.e., the underlying metrics such as success rate; while $\tau(n_i, n_j)-$ denotes punishment. The difference between proxy- and triad pattern is the establishment of a relationship between the original requester r and the actually serving s_x.

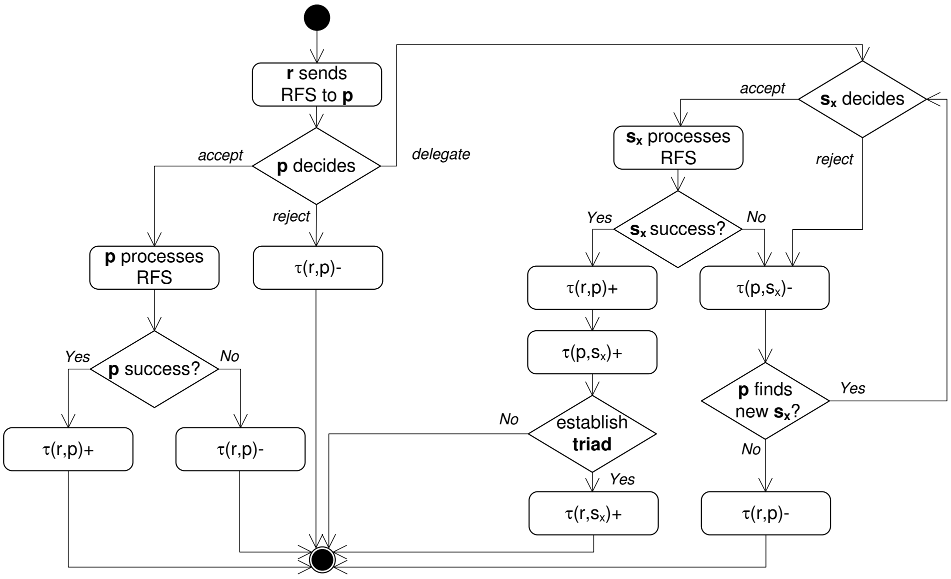


Figure 5.7: Rewarding and punishment of trust relations according to delegation behavior.

After evaluation of rewarding rules, the system updates confidence values and their reliability in the corresponding trust scopes. Besides automatically determined metrics, we incorporate a manual rewarding mechanisms (feedback provided by users). Both, automatic and manual rewards, are equally weighted and merged (averaged), to extend the dimensions of trust, and strengthen its information value for the users of the Expert Web.

5.6 Discussion

In this chapter, we introduced concepts, centered around trust and Human-Provided Services in mixed service systems, and discussed their application in context of the *Expert Web* enterprise use case. We introduced models for rewarding interactions and establishing trust on top of delegation patterns that typically occur in real-world scenarios. Besides the detailed use case and application of trust in enterprise networks, we demonstrated the technical realization of our approach, focusing implementation details that apply ECA rules.

Evaluations of delegation, rewarding and trust inference concepts can be found in the next chapters of this work. However, further research has to be conducted to make the presented concepts applicable in more complex environments.

Part II

Trust Mining and Prediction on the Web

CHAPTER 6

Trust and Reputation Mining in Professional Virtual Communities

Outline. In the case there are no observable SOAP interactions, trust is going to be based on other data, gathered through mining on the Web. We present an approach to determining relations of users in online discussion forums.

Contents

6.1 Motivation

The concept of virtual (or online) communities is quite common today and frequently used not only for private concerns, but also in professional working environments. Online platforms such as discussion forums, blogs, and newsgroups are regularly utilized to get introduced into new topics, to find solutions for particular problems, or just to stay informed on what's up in certain domains. Virtual communities are rapidly growing and emerging, and thus, lots of spam and dispensable comments are posted in their forums or sent via e-mail, polluting fruitful discussions. Several mechanisms have been proposed to handle this problem, such as collaborative filtering of comments and global reputation of users based on feedback mechanisms. However, because these concepts rely on manual and subjective human feedback, they suffer from several drawbacks [58], including unfair ratings, low incentives for providing feedback, and quality variations of ratings over time.

Especially, where mentioned communication technologies are regularly embedded to connect e-professionals, such as in professional virtual communities (PVCs), and where successful collaboration is critical for business, we identified the need for more sophisticated reputation methods. Moreover, in modern working environments, where virtual teams consisting of members from different departments or companies work together, personally unknown to each other, various complex social factors affect the overall collaboration success. These factors can be expressed by one composite and abstract concept: *trust*. Trusted relationships between colleagues are vital to the whole collaboration and a prerequisite for successful work. A recent report about the roles of trust in today's business world [122] discovers that besides professional skills expressed as experience, expertise and competence, soft skills, such as the willingness to exchange information, motivation and communication skills, are at least equally important. Such social skills can be discovered and evaluated in typical computer-supported discussions, common in online communities, including threaded forum discussions, instant messaging chats, and e-mail conversation.

In this chapter, we deal with the following contributions:

- *Concept of Trust Mining.* We show an approach to trust mining between discussion participants. Here, trust reflects someone's ability to serve as valued discussion partner in certain topics.

- *Mining Algorithm.* Our main contribution is a mining algorithm that is applied in threaded Internet forums.

- *Evaluation and Discussion of Trust Mining.* We verify our approach with real data sets and discuss its application and gathered results.

Trust mining is of paramount importance if interactions cannot be directly observed due to technical or legal reasons. Then, trust mining deems to be a convenient way to determine relations between users through processing interaction data that is accessible from the Web.

6.2 Trustworthy Sources of Data

Most common online communication platforms, such as vBulletin[1], integrate reputation systems which either rank users based on simple metrics, including their posting count, or enable users to reward ('thank') others directly. In contrast to this approach, we reward the *vitality of relationships* between users and then derive user ratings by aggregating relationship ratings. This enables us to utilize global metrics calculated from all available data, such as the overall discussion effort of a particular user with respect to the whole community, but also local metrics considering data restricted to particular users only, such as the discussion effort between two specific users. Utilizing local metrics is of particular interest

[1] `http://www.vbulletin.com`

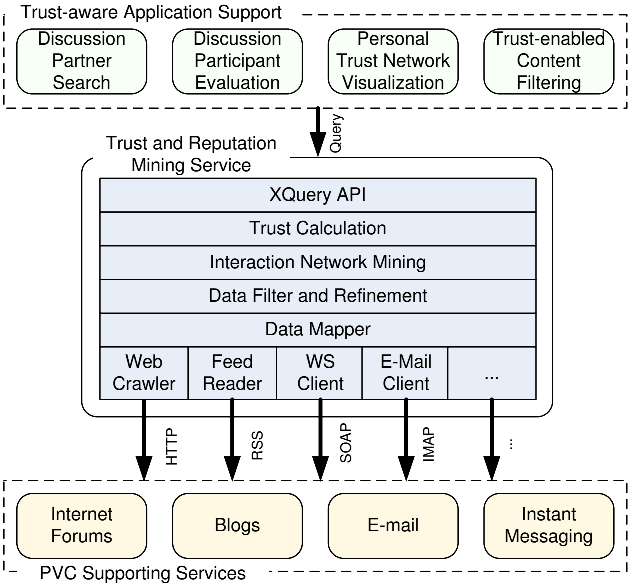

Trust-aware Application Support
Discussion Partner Search
Discussion Participant Evaluation
Personal Trust Network Visualization
Trust-enabled Content Filtering
Query
Trust and Reputation Mining Service
XQuery API
Trust Calculation
Interaction Network Mining
Data Filter and Refinement
Data Mapper
Web Crawler
Feed Reader
WS Client
E-Mail Client
...
HTTP
RSS
SOAP
IMAP
...
Internet Forums
Blogs
E-mail
Instant Messaging
PVC Supporting Services

Figure 6.1: Architectural overview of the trust and reputation mining service.

when the amount of controversial users is high [77]. With our system a user does not only receive one globally valid trust rank, but may be graded from each individual's view.

We developed a pluggable architecture (Figure 6.1) - part of VieTE [120] - that utilizes various communication data sources through standard protocols, such as RSS feeds, SOAP, and e-mail. The obtained data is mapped to a generic communication schema, pre-processed and refined. Finally, an interaction network that models the relationships between users upon mining communication logs, is built. Based on this network, trust between individuals and reputation from a (sub-)community's perspective can be inferred and queried through VieTE's provisioning interface. We understand the inferred *discussion trust* to represent one dimension of general trust in PVCs, applicable in a wide range of differently organized communities. Other dimensions of trust may base on the fulfillment of service level agreements or the reliability of task execution, which are in scope of previous chapters.

6.3 Trust and Roles in Virtual Community Discussions

In discussions we can intuitively distinguish between information providers and information consumers. Especially in online discussions we can easily track who provides information, e.g., by posting a comment in a forum or writing an e-mail to a group of people. In contrast to that, determining information consumers is tricky. We can never be sure, that people read received e-mails or new comments in a forum, even when they open forum entries in their browsers. However, if somebody replies to a particular comment, then we can certainly assume, s/he has read the message and found it worth for discussion. Thus the replier can be identified as an information consumer, but also as an information provider.

In our approach we track exactly this discussion behavior and define, that whenever one replies to a comment of another one, an interaction between them takes place. We destill these interactions and build a notion of trust on top.

Particularly in discussions, it may seem intuitive that the more comments somebody provides the more s/he can be trusted to be a good discussion partner. However, *lurkers* [91], referring to people just watching discussions but not actually participating, can be less trusted regarding their 'openness' and active participation. They lack the willingness to exchange information, motivation or communication skills, thus they are bad collaborators.

However, a simple comment count does not truly reflect if somebody's statements are real contributions and worth reading and discussing. Thus, we consider threaded structures and analyze how comments are recognized by others. For that purpose, we define the following novel social roles within discussion scenarios:

- *Activator.* The role of an activator reflects that the more replies a discussion participant receives, the more one's comments seem to to be worth for discussion. Thus, one can be trusted to have the competencies and skills to provide comments, interesting for a wide range of community members.

- *Driver.* The role of a driver reflects, the more somebody replies to comments, the more s/he can be trusted to actively participate in a discussion, thus s/he represents a catalyst evident for a fruitful discussion.

- *Affirmed Driver:* An affirmed driver is defined as a driver whose contribution is affirmed. This is the case if there is at least one reply to a driver's comment.

According to these roles, *discussion trust* is (i) a measure for the contribution to discussions expressing the willingness to provide information and support; (ii) a measure for interest similarities of people, as frequent discussion partneres have stronger relations. Note, discussion trust does not reflect that a particular participant offers a valid information or posts the truth. For this purpose, natural language processing and analyzing semantic meanings of comments are required [33, 126], which is out of scope of our work.

6.4 Discussion Mining Approach

We developed a mining algorithm to determine the contribution of people in discussions. However, in contrast to common approaches, we neither reward the participants directly (e.g., their number of provided comments), nor do we utilize subjective feedback. We rather mine interactions to reward particularly the relationships between each pair of discussion participants.

We make the following assumptions: (i) The notion of time can be neglected, which means our algorithms do not determine how trust relations change over time. We determine trust relations for one particular point in time through mining of short history data. Temporal evaluations, e.g. by applying moving averages, temporal weighting functions or sliding windows, have to be set up on top of our approach and has been discussed in previous chapters. (ii) We do not apply natural language processing. Thus, we accept the introduction of noise and small errors by rewarding users who post useless comments (i.e., spam). In the evaluation part we show that this is no disadvantage if we rely on larger amounts of data. We further assume that in PVCs spam occurs less frequently than in open Internet forums.

6.4.1 Interaction Network Definition

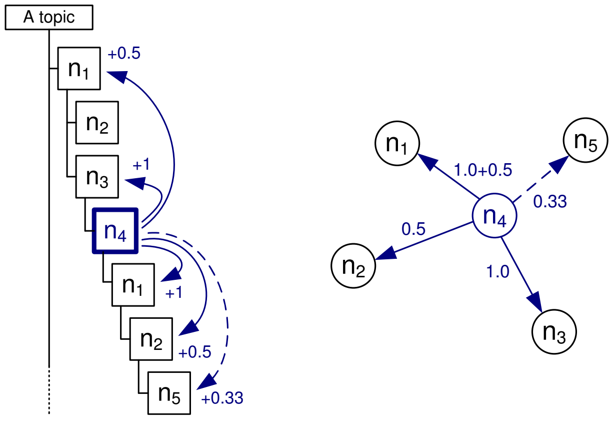
Again, we utilize a directed graph model $G = (N, E)$ to reflect discussion relationships E between users $n_i \in N$, and incorporate context to allow trust determination with respect to different situations on top of the created interaction network. A relationship $e(n_1, n_2) \in E$, as defined in Equation 6.1, is described by various $metrics$ such as the number of recent interactions, their weights, and communication scores, valid in particular scopes (described by context elements).

$$e(n_1, n_2) = \langle n_1, n_2, metrics[name, value, scope] \rangle \tag{6.1}$$

6.4.2 Discussion Mining Algorithm

We develop an algorithm that weighs the communication relations based on discussions between each pair of participants. Let us assume an environments supporting threaded discussion structures, as common in online forums or newsgroups. We argue that somebody who provides a comment in a discussion thread is not only interested in the comment s/he directly replies to, but to a certain extent also by preceding posts in the same chain of comments. Thus, we interpret a thread to be similar to a group discussion and establish relationships between participants who are posting in one chain. Figure 6.2(a) shows a structured discussion thread where every box represents a comment provided by the annotated participant. For the highlighted comment provided by n_4, arrows show exemplary which interactions between participants are synthesized by our algorithm. The comment provider n_4 honors the attracting comments of n_3 and n_1, and rewards the driving contributions of n_1, n_2, and n_5. If only affirmed drivers shall be rewarded, then the relation to n_5 (dashed lines) is skipped, because no one has been attracted by its comment. The weights of interactions are calculated by the interaction reward function $f_i(dt, c1, c2)$, where D is the discussion tree, and the interaction from the author of comment c_1 to the author of c_2 is rewarded. We initially set $f_i(D, c_1, c_2) = \frac{1}{dist(c_1,c_2)}$, where $dist()$ determines the distance between two comments, i.e., the number of intermediate posts in the same thread (direct replies have $dist = 1$). However, considering further comment attributes, including time intervals between a comment and its replies or the number of replies a single comment attracts, may improve the expressiveness regarding trust. All interactions between two particular participants are aggregated and directed weighted relations are created in the graph model shown in Figure 6.2(b).

Algorithms 4 and 5 describe formally the mode of operation. According to Equation 6.1, each edge in the interaction model can have various metrics. Currently we apply *count*, which is the amount of interactions between two participants, and *strength*, which is the sum of the weights of all interactions between them. We utilize the function



(a) Discussion thread

(b) Mapping for user n_4

Figure 6.2: Mapping from a discussion thread to the interaction network model.

`incMetric(name, scope, edge, value)` to increment the metric in `scope` specified by `name` of the given `edge` by a certain `value`.

In Algorithm 4 relations from a comment's c provider to the providers of preceding comments are established due to their activator role. Algorithm 5 establishes relations to the providers of child comments due to driving behavior. The function `provider()` returns the identity of a comment provider, `parent()` determines the parent comment on the specified level of the discussion thread ($lvl = dist(c1, c2)$), and `children()` provides child comments.

Algorithms 4 and 5 are applied for each comment and reward the comment provider's contribution to the overall discussion. The approach can be further improved by accounting for frequent communication patterns. This means, if n_i provides a comment replied by n_j,

Algorithm 4 Function for rewarding the relations to the activators of a comment c

```
Require: discussionThread $D$, graphModel $G$, comment $c$, Scope $s$
/* reward postings across the number of configured levels */
for $lvl$ = 1 to configMaxLevelUp do
    $c_p \leftarrow$ parent($D, c, lvl$)
    /* break on top comment and do not reward self-replies */
    if $\nexists$ $c_p$ or provider($c_p$) = provider($c$) then
        break
    /* create edge on demand */
    if $\nexists$ edge($G$, provider($c$), provider($c_p$)) then
        createEdge($G$, provider($c$), provider($c_p$))
    /* reward activator on current $lvl$ */
    incMetric($G$, strength, edge(provider($c$), provider($c_p$)), $s$, $1/lvl$)
    incMetric($G$, count, edge(provider($c$), provider($c_p$)), $s$, 1)
    $lvl \leftarrow lvl + 1$
return $G$
```

Algorithm 5 Function for rewarding the relations to the drivers of a comment c

```
Require: discussionThread $D$, graphModel $G$, comment $c$, Scope $s$
/* reward postings across the number of configured levels */
for $lvl$ = 1 to configMaxLevelDown do
    $C_c \leftarrow$ children$(D, c, lvl)$
    /* bottom level of $D$ reached */
    if $C_c = \emptyset$ or then
        break
    /* reward all driving child comment providers */
    for all $c_c \in C_c$ do
        /* do not reward self-replies */
        if provider($c_c$) = provider($c$) then
            break
        /* create edge on demand */
        if $\nexists$ edge($G$, provider($c$), provider($c_c$)) then
            createEdge($G$, provider($c$), provider($c_c$))
        incMetric($G$, strength, edge(provider($c$), provider($c_c$)), $s$, $1/lvl$)
        incMetric($G$, count, edge(provider($c$), provider($c_c$)), $s$, 1)
    $lvl \leftarrow lvl + 1$
return $G$
```

Algorithm 6 Function for rewarding bidirectional communication

```
Require: discussionThread $D$, graphModel $G$, comment $c$, Scope $s$
/* determine $c$'s parent and child comments */
$c_p \leftarrow$ parent$(D, c, 1)$
$C_c \leftarrow$ children$(D, c, 1)$)
/* check for bidirectional communication */
for all $c_c \in C_c$ do
    if provider($c_p$) = provider($c_c$) then
        incMetric($G$, strength, edge(provider($c$), provider($c_c$)), $s$, $bidiR$)
return $G$
```

and n_i replies to n_j's comment, then a real bidirectional communication can be observed. In this case, the metric *strength* of $e(n_j, n_i)$ is additionally rewarded with $bidiR$, because n_i does not only provide comments recognized by n_j, but n_i also answers to n_j's replies (see Algorithm 6).

6.5 Trust Mining Model

6.5.1 Trust Inference

Similar to our previous approaches and other work [12, 55], trust is determined on top of the created interaction network, depending on the notions of confidence and reliability. We define that the confidence of user n_i in user n_j with respect to scope s (e.g., the domain of discussion) can be derived from the previously described graph model G by using a confidence function $c^s(n_i, n_j) = f_c(G, n_i, n_j, s)$.

Reliability, expressing the certainty of n_i's confidence in n_j with respect to scope s, is determined by a reliability function $\rho(c^s(n_i, n_j)) = f_\rho(G, n_i, n_j, s)$. The value of $\rho(c^s(n_i, n_j)) \in [0, 1]$ is basically influenced by the number and type of interactions which were used to calculate confidence, and expresses the reliability of the confidence value between totally uncertain and fully affirmed.

With the confidence of n_i in n_j and its reliability we calculate trust $\tau^s(n_i, n_j)$ of n_i in n_j according to Equation 6.2.

$$\tau^s(n_i, n_j) = c^s(n_i, n_j) \cdot \rho(c^s(n_i, n_j) \tag{6.2}$$

6.5.2 Trust Aggregation and Reputation

Aggregation of trust, previously referred to as reputation, refers in this regard to (i) the composition of trust values of a group of users in one user to build a view of trust from a community's perspective, or (ii) the composition of trust values calculated for different scopes between two users to get a notion of trust for a broader scope or (iii) the combination of (i) and (ii) to get the 'general' community trust in one user.

The computation of trust follows our introduced model in Chapter 5. Equation 6.3 is applied to determine reputation tau_{rep} of a group $N' \subseteq N$ of users in one particular user $n_j \in N'$ with respect to a set of scopes S. The weighting factor calculated by f_a can be configured statically or set dynamically depending on individual properties of elements in N', e.g., trust of long-term users have a higher impact on reputation than those of newbies.

$$\tau_{rec}^S(n_j) = \frac{\sum_{n_i \in N'} \sum_{s \in S} \tau^s(n_i, n_j) \cdot f_a(G, n_i, n_j, s)}{\sum_{n_i \in N'} \sum_{s \in S} f_a(G, n_i, n_j, s)} \tag{6.3}$$

6.6 Evaluation and Discussion

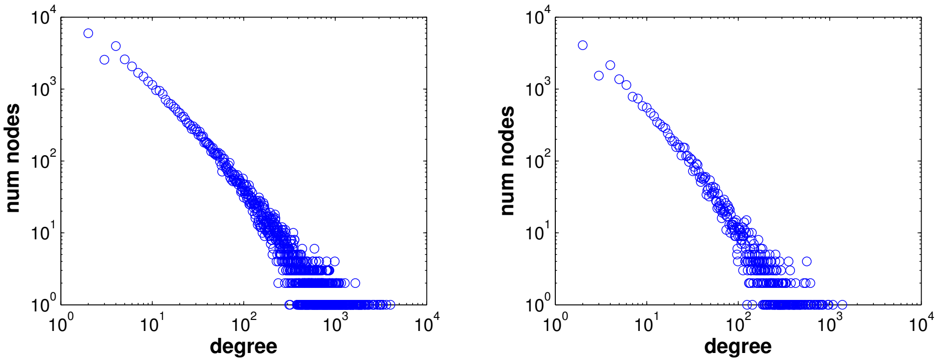
6.6.1 Preparing Evaluation Data

For the evaluation of our approach, we compare the output of the proposed algorithm with real users' opinions. Because our developed system is new and currently not utilized by a wide range of users, we need a dataset which offers structured discussions in various contexts and information about the real contribution of users. We fetched an appropriate dataset with the required characteristics from the famous Slashdot[2] portal.

Slashdot is a platform which offers the ability to discuss a wide variety of topics classified in different subdomains. An additional feature is the moderation system allowing experienced users to rate the postings of other users on a scale between -1 and 5. We interpret this score as human feedback which provides information about the quality of comments and thus, when considering all posts, the average discussion capabilities of a person.

We developed a Web crawler to capture threaded discussions in the subdomains *Your Rights Online (yro)* and *Technology (tech)* from January 2007 to June 2008. We selected

[2] `http://slashdot.org`



(a) Connection degree (full data).

(b) Connection degree (reduced data).

Figure 6.3: Degree distribution.

these two subdomains due to their diversity, expressing different interests and expertises of people discussing there. The subdomain in which a discussion takes place is reflected by the scope of a discussion: s=$\{yro \mid tech\}$. Users may have established discussion relationships with respect to either yro, or $tech$, or both.

We have to ensure to compensate all impacts that degrade the quality of the data set and suitability for the tests. First, we remove all comments posted by anonymous users, because there is no meaningful way to map this data to particular nodes of the interaction graph model. Second, if not changed from the default settings, the Slashdot UI hides low scored comments automatically. Therefore, there is no way to distinguish if a particular comment is not replied because it is simply poor and not worth a discussion, or if it is not replied because it is hidden and thus never read. Hence, we remove low scored comments from the data set. Third, we remove all potentially unrated posts, i.e., posts with score 1, to ensure unbiased experiments.

Initially the captured data set consists of 49.239 users and 669.221 comments in the given time period. After applying all steps of reduction we map the discussions to the graph model, consisting of 24.824 nodes and 343.669 edges. In the experiments we rank each user relatively to the others based on how much their discussion skills can be trusted by the rest of the community. Because our presented trust mining approach fully relies on the connectivity of a node within the graph, we have to ensure that the filtering procedures do not distort this property. Figure 6.3 shows the degree of connection for each node for the full data set and for the reduced one. The distribution follows a common power law function, and when applying the reduction steps, the characteristics of the user distribution and their connectivity basically do not change.

6.6.2 Trust Network Model Configuration

By applying the presented mapping approach we are able to grade discussion relationships between any two users n_i and n_j in the graph $G = (N, E)$ with respect to the subdomain, reflected by scope s=$\{yro \mid tech\}$.

Trust is determined by confidence and reliability as described in Section 6.5. To this end, we define $f_c(G, n_i, n_j, s) = \text{metric}(strength, e(n_i, n_j), s)$ to be a function which simply returns the discussion strength of the relation from n_i to n_j in a specific subdomain (= scope s). We define a notion of confidence from n_i in n_j to be fully reliable if there are at least max_{ia} interactions with respect to the same subdomain. If $f_\rho(G, n_i, n_j, s) = \frac{\text{metric}(count, e(n_i, n_j), s)}{max_{ia}}$ is greater than 1 we set $f_\rho(G, n_i, n_j, s) = 1$. We configure $max_{ia} = 10$ per year, which is the same amount of posts as identified in [42] to be required to calculate representative results. For computing reputation, we apply all single input trust values having the same weight $f_a(G, n_i, n_j, s) = 1$.

For the sake of clarity we apply only the simple functions defined above, however, more complex functions can be set up, which consider similarities between subdomains, the amount of interactions compared to well-known community members or symmetry of trust relationships, just to name a few.

Furthermore, we set $configMaxLevelUp = 3$, $configMaxLevelDown = 3$ and reward bidirectional communication, i.e., post-reply-post patterns, with $bidiR = 1$ extra point. By further increasing the number of levels for rewarding, the values indicating discussion strength between the users will increase as well. However, this does not highly influence the relative rankings of users.

6.6.3 Evaluation Approach

We evaluate our trust mining algorithm approach by comparing its results with trust values derived from the feedback of real users. We introduce the following terminology:

> *Link rank.* The link rank of a user is calculated by our mining algorithm accounting for the strength of connections to others based on their nested comments within discussions. We interpret this measure as trust and argue, that it directly reflects a user's willingness to share information and support others (driver role), and attitude to highly recognized contributions (activator role).

> *Score rank.* The score rank of a user is calculated by averaging his/her posting scores, thus we utilize direct human feedback. We interpret the score rank as trust and argue, that users may trust posters with high average posting score more to deliver valuable contributions, than others.

Obviously both ranking methods rely on the same social properties, which reflect the value of contribution provided by community members.

Note, our proposed scoring method does not only depend on the number of posts and is completely different from simply giving reward points for every posted comment such as in common Internet forums. Figure 6.4 depicts the number of posts within 18 month of the top1000 linked users. However, there is a trend that frequently posting users are ranked higher, there is obviously no strong correlation between the link rank and the number of posts.

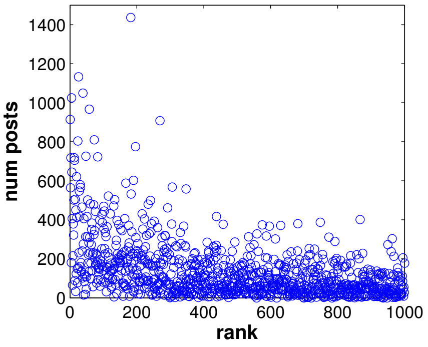


Figure 6.4: Link rank compared to number of posts for top1000 linked users.

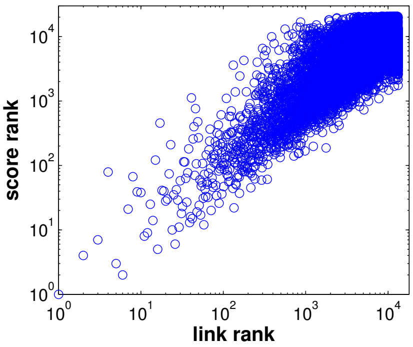


Figure 6.5: Link rank compared to score rank for each user.

6.6.4 Experiments

6.6.4.1 Calculating Global Reputation

In our first experiment we determine global link ranks that are built by aggregating the link strength values of all individual relations within the network for each user over all scopes. Besides this, we determine the global score rank as well. This means we rank each user two times: once with our algorithm through mining discussion structures, and once based on humans' feedback scores. For determining score ranks we degrade users' average scores by the factor $\frac{postcount}{numMinposts \cdot numMonth}$, if they posted less than $numMinposts$ posts a month. This ensures that rarely posting users with highly rated comments do not outrank frequently posting users. For instance, a user with 5 comments, all rated with the highest possible score, should not outrank another user with 100 posts with only 98 comments that are scored highest. During experiments we found out that $numMinposts = 10$ per month seems to be the value to reach the highest Pearson correlation coefficient (0.77) between the results of both ranking methods for the given data set, as shown in Figure 6.5.

We further calculate the Dice similarity coefficient depicted in Equation 6.4, which is defined as the amount of elements included in both of two sets, in our case the sets of top scored users (TopXS) and top linked users (TopXL), where X={10,25,50,100,1000} determining the size of the sets.

$$s = \frac{2 \cdot |\text{TopXS} \cap \text{TopXL}|}{|\text{TopXS}| + |\text{TopXL}|} \tag{6.4}$$

Table 6.1 shows how many percent of top linked users and top scored users overlap after different time intervals. Obviously, the more data is used for trust calculation the more the resulting top linked users get similar to the top scored ones, which means we receive preciser results. After 18 month we finish with an overlap between 45 and 60 percent, for the top10 to top50 and approximately 65 to 70 percent for larger groups. Furthermore, we compare the amount of the top10 scored (Top10S) users who are also in the top25, top50, top100, and top1000 (TopXL) of the top linked users. The top10 scored users are the users scored best by others, and thus are most trusted to provide meaningful information. Table

6.1 shows that after 4 month 90 to 100 percent of the top10 scored users are included in the top50 linked users.

OSim after month:	01	02	03	04	06	10	14	18
Top10|TopS10 in TopL10	10|10	30|30	30|30	30|30	40|40	50|50	60|60	50|50
Top25|TopS10 in TopL25	32|50	36|40	48|70	60|80	52|80	48|70	44|70	44|90
Top50|TopS10 in TopL50	28|50	34|60	40|80	50|90	54|100	58|90	62|100	60|100
Top100|TopS10 in TopL100	36|90	42|90	46|90	48|100	58|100	66|100	70|100	64|100
Top1000|TopS10 in TopL1000	61|100	61|100	66|100	64|100	64|100	66|100	68|100	70|100
number of users x1000	2.5	4.9	6.4	7.9	11	15	18	20

Table 6.1: Overlap similarities (OSim) of top linked and top scored users in percent.

We conclude, that for the given data set we are able to find a similar set of users, who are trusted to post high quality comments, when ranked either by the average of posting scores (scoreRank) or by the discussion structure and reply behavior (linkRank).

6.6.4.2 Enabling Context Dependent Trust Ranking

In a second experiment we consider the discussion scope. Discussions in the utilized dataset take place either in subdomain `yro` or `tech`. We show that it is reasonable to calculate trust for particular situations reflected by scopes. We use six month of data from January 2008 to July 2008 because in this interval the amount of discussions and user distribution in both subdomains are nearly equal, so, results cannot be influenced by the number of posts. Then we rank each user two times with our algorithm, once for discussions in `yro` and once for `tech`. We rank only users with more than 10 posts, which we defined earlier as the absolute minimum for computing reliable results. There are in sum 14793 different users, where 5939 are only active in `yro` and 6288 in `tech`. Other users participate in discussions in both subdomains and thus, are ranked two times.

In Figure 6.6 we compare how users are ranked with respect to both subdomains. There is an amount of approximately 40 users who are both, in the top100 wrt. `yro` and in the top100 wrt. `tech`, hence these people are highly trusted independent from the subdomain. However, there are around 60 users in the top100 of one subdomain but badly ranked in the other one, or not participating in discussions in the other subdomain at all. This group is reflected in Figure 6.6 in the top-left quadrant for `yro` and in the bottom-right for `tech` respectively.

We conclude that between the sets of top100 trusted users with respect to each subdomain there is less overlap than diversity. These results show the usefulness of considering trust scopes.

6.6.4.3 Determining Personal Trust

In contrast to reputation, which is mostly defined as the aggregated opinions of others, trust relies on personal experiences. As described in [42], in typical online communities exist several clusters of users that comprise tightly interconnected members, but only sparse connections to other clusters.

Compared to most common reputation systems that maintain only one global rank, e.g., reputation for each user from a global point of view, we are able to determine personal trust

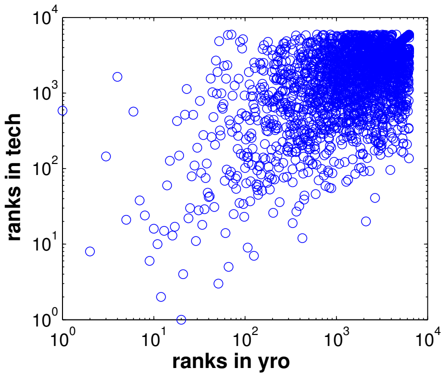


Figure 6.6: Link ranks in different contexts.

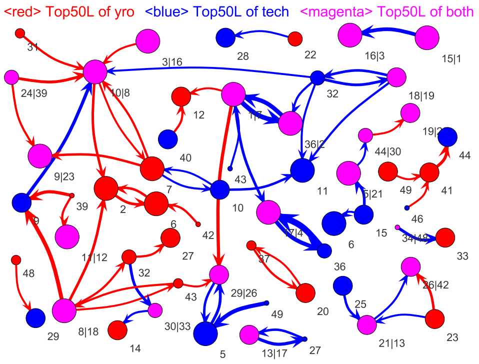


Figure 6.7: Trust network (reduced).

relations (from an individual's view). Hence, each community member has three possibilities to determine trustworthy discussion partners: (i) trust users with highest reputation from a global view (with or without distinguishing scopes), (ii) trust users who are directly connected strongest by utilizing local metrics (however, these users may have only an average global reputation) or (iii) combine both methods.

In Figure 6.7 we removed all connections with strength ≤ 5, and all users who are either not in the top50L users of *yro* (red), *tech* (blue), or both (magenta), or not connected to anyone else. Therefore, the users with the highest reputation and their strongest connections remain. The size of a node (users) depends on its global rank in either *yro*, *tech* or both, and the width of an edge reflects the connection strength. Obviously the trust graph splits into several only sparsely interconnected components.

We investigated the connectivity of users to top users in the graph model. The subset of users who are both connected to at least two other users and having more than 10 posts a month has a size of 2038. Nearly all users (2014) are directly connected to at least one member of the top250 linked users. At least half of them (1080) are connected to one user from the top100, but only around 10% of all users are connected to a top10 one.

However a large amount of users is directly connected to at least one globally top ranked discussion participant, there is still the need to account for personal relationships. Especially for the presence of controversial users [77], personal relations are the only reliable way to determine trusted discussion partners from the individuals' views.

CHAPTER 7

Bootstrapping and Prediction of Trust

Outline. This chapter deals with bootstrapping, i.e., predicting, trust between actors in the case no interactions from previous collaborations have been captured.

Contents

7.1 Motivation

Trust and reputation mechanisms are essential for the success of open, large-scale Web systems. In such systems, usually information provided by users or obtained during their interactions, is collected to detect beneficial social connections, potentially leading to trust between their members. While many trust-, recommendation- and reputation systems have been described, including their underlying models and modes of operation [44, 58, 100], one particular problem has been mostly neglected: *How to put the system into operation*, i.e., how to bootstrap trust between users to let them benefit from using the system, even if there is not yet much, or even no, collected data available. A further, closely related research questions is, *how to integrate newcomers that have no relations to any other members in the system*.

Our prior work [111] describes an environment comprising humans and services, in which interactions spanning both kinds of entities are monitored. We strongly believe that trust can only be based on the success and outcome of previous interactions [111, 119]. Without having knowledge of prior (observed) interactions, we argue that trust between users cannot be determined in a reliable manner. Therefore, we propose an approach for *trust prediction* that aims at compensating the issue of bootstrapping trust. We consider influencing factors stimulating the evolution of trust. In various environments, such as collaborative systems, trust is highly connected to interest similarities and capabilities of the actors. For instance, if one actor, such as a human or service, has the capabilities to perform or support a collaborative activity reliably, securely and dependably, it may be sensed more trustworthy than other actors. Moreover, we argue that if actors have interests or competencies similar to well-known trusted actors, they may enjoy initial trust to some extent.

The contributions of this chapter are as follows:

- *Application Environment.* We introduce our concepts to trust prediction, and model an application environment that enables trust prediction, even in the absence of direct interactions.
- *Bootstrapping Mechanisms.* We present our approach for creating and comparing tagging profiles based on clustering, and a novel method for trust prediction using similarity measurements.
- *Evaluation and Discussion.* We show a reference implementation of our approach, and evaluate algorithms using real world data sets from the tagging community `citeulike`[1].

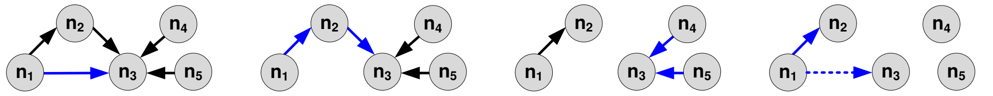
7.2 Towards Prediction of Trust

Trust between entities can be managed in a graph model (for example, see [7]). The graph is defined as $G = (N, E)$ composed of the set N of nodes defining actors trusting each other and the set E of directed edges denoting trust relations between actors. This model is known as the *Web of Trust*.

In Figure 7.1, four different scenarios are depicted, which show concepts for trust determination in a Web of Trust. We assume a situation where trust from actor n_1 to actor n_2 is to be determined. The first case in Figure 7.1(a) visualizes the optimal case, in which a trust relation from n_1 to n_2 can be inferred directly, e.g., based on previous interactions [111]. In the second case in Figure 7.1(b), no direct trust relation could be determined, however trust can be propagated if we assume transitivity of trust relations [58], enabling n_2 to recommend n_3 to n_1. The third case in Figure 7.1(c) depicts, that there is neither a direct nor a propagated trust relation from n_1 to n_3. However, unrelated third party actors n_4 and n_5 may provide a weaker, but acceptable, notion of trust in c through the means of reputation. For our work the fourth use case in Figure 7.1(d) is the most interesting one,

[1] `http://www.citeulike.org`

which demonstrates the limitations of propagations in the Web of Trust. If no one interacted with n_3 in the past and no one has established trust in n_3, new approaches to trust prediction need to be applied.



(a) Direct trust inference from n_1 to n_3.

(b) Recommendation of n_3 from n_2 to n_1.

(c) Reputation of n_3 by n_4 and n_5.

(d) Similarity of (n_1, n_3) and (n_2, n_3) for trust prediction.

Figure 7.1: trust(n_1, n_3)=?: The need for trust prediction in a Web of Trust.

We distinguish the following both trust prediction concepts:

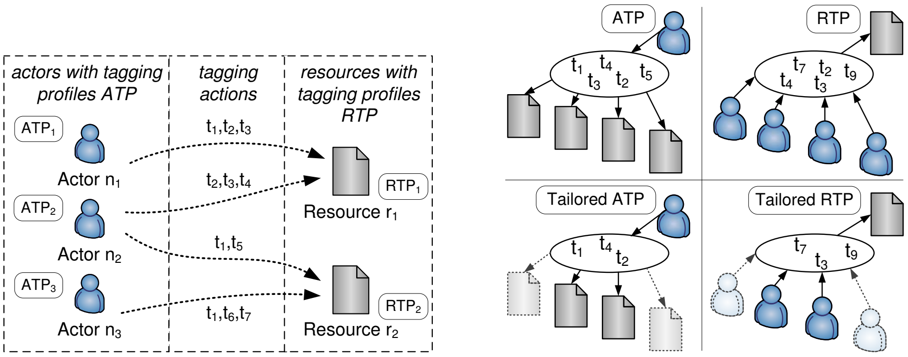
- **Trust Mirroring.** Depending on the environment, interest and competency similarities of people can be interpreted directly as an indicator for future trust. This is especially true in environments where actors have the same or similar roles (e.g., online social platforms). There is strong evidence that actors who are 'similar minded' tend to trust each other more than any random actors [78, 134]; e.g., movie recommendations of people with same interests are usually more trustworthy than the opinions of unknown persons. In Figure 7.1(d), this means measuring the similarity of n_1's and n_3's interests, allows, at least to some extent, trust prediction between them (dashed line).

- **Trust Teleportation.** As depicted by Figure 7.1(d), we assume that n_1 has established a trust relationship to n_2 in the past, for example, based on n_2's capabilities to assist n_1 in work activities. Therefore, others having similar interests and capabilities as n_2 may become similarly trusted by n_1 in the future. In contrast to mirroring, trust teleportation is applied in environments comprising actors with different roles. For example, a manager might trust a developer belonging to a certain group. Other members in the same group may benefit from the existing trust relationship by being recommended as trustworthy as well. We attempt to predict the amount of future trust from n_1 to n_3 by comparing n_2's and n_3's interests and capabilities.

Sophisticated profile similarity measurements are needed in both cases to realize our concepts of trust prediction.

7.3 Tagging Environment

According to our concepts of trust prediction, we need models to manage the interests and competencies of humans, and features of resources, e.g., services, respectively. In contrast to traditional centralized approaches, where one instance, such as the human resource department, manages a catalog of competencies, we follow a *dynamic self-managed user-centric approach*. We assume an environment where each actor tags different types of resources s/he is interested in, such as bookmarks, scientific papers and Web services.



(a) Actors tagging resources.

(b) Different types of tagging profiles.

Figure 7.2: Description of the tagging environment.

Based on the type of tagged resource and assigned tags, we can infer the centers of interest, expressing to some extent their knowledge areas and capabilities; but from a community's view also the features or preferred usage of tagged resources. By utilizing this knowledge and applying our concepts of *trust mirroring* and *trust teleportation*, we think it is possible to predict trust relations that potentially emerge in the future.

We model the environment as depicted in Figure 7.2(a) which consists of:

- a set of actors N, having different interests reflected by actor-tagging-profiles (ATP). These profiles are derived from tags $T' \subseteq T$ used by $n_i \in N$ on a subset of resources $R' \subseteq R$.
- a set of resources R, having different properties (covering actor interests) reflected by resource-tagging-profiles (RTP). These profiles are derived from tags $T' \subseteq T$ used by a subset of actors $N' \in N$ on $r_j \in R$.
- a set of tagging actions $T = \{t_1, t_2, t_3 \dots\}$, where each t_x is created by an actor $n_i \in N$ for a resource $r_j \in R$.

7.3.1 Modes of Profile Similarity Measurement

We determine *tagging profiles* for both actors (ATP) and resources (RTP) (Figure 7.2(b)). ATPs express independent from particular resources, which tags are frequently used by actors and therefore, their centers of interest. RTPs describe how a certain resource is understood in general, independent from particular actors. According to our motivating scenario depicted in Figure 7.1(d), ATP similarities can be either interpreted in context of *trust mirroring* or *trust teleportation*. In contrast to that, RTP similarities are mostly only meaningful for *trust teleportation* (e.g., Actor n_i trusts n_2 who is interested in or familiar with a resource r_j, thus n_i might trust n_3 who uses a very similar resource r_k as well.)

Compared to general profile similarity, and common profile mining approaches, e.g. in recommender systems [108], we do not only capture which actor uses which tags (ATP) or which resource is tagged with which tags (RTP). We rather consider how an actor tags particular subsets of resources. Using such *Tailored ATP*s we can infer similarities of tag usage between actors $n_i, n_j \in N$, and therefore similarities in understanding, using, and apprehending the same specific resources $R' \subseteq R$. Furthermore, we capture how two resources $r_i, r_j \in R$ are tagged by the same group of actors $N' \subseteq N$. Such *Tailored RTP*s can be utilized to determine similarities between resources and how they are understood and used by particular groups of actors; e.g., distinguished experts in their respective fields (Figure 7.2(b)).

7.4 Similarity-based Trust Prediction

Similarities of actors' tag usage behavior can be directly calculated if an agreed restricted set of tags is used. There are several drawbacks in real-life tagging environments that allow the usage of an unrestricted set of tags. We identified two major influencing factors prohibiting the direct comparison of tagging actions. First, synonyms cause problems as they result in tags with (almost) the same meaning but being differently treated by computer systems, e.g., `football` v.s. `soccer`. Second, terms, especially combined ones, are often differently written and therefore not treated as equal, e.g., `social-network` v.s. `socialnetwork`.

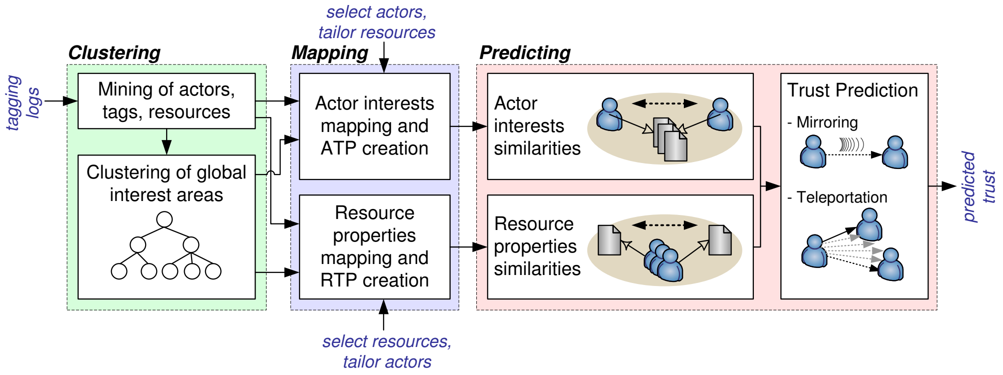

select actors, tailor resources
Clustering
Mapping
Predicting
tagging logs
Mining of actors, tags, resources
Clustering of global interest areas
Actor interests mapping and ATP creation
Resource properties mapping and RTP creation
Actor interests similarities
Resource properties similarities
Trust Prediction
- Mirroring
- Teleportation
predicted trust
select resources, tailor actors

Figure 7.3: An approach to trust prediction based on clustering and similarity.

Due to the described drawbacks of comparing tagging actions directly, we developed a new approach, which measures their similarity indirectly with respect to a global reference model. This approach to similarity measurement and *trust prediction*, is depicted in Figure 7.3. Three steps are performed:

1. *Clustering.* Identifying tagging actions, each consisting of an actor $n_i \in N$ tagging a resource $r_j \in R$ using tags $T_{r_j} = \{t_1, t_2, t_3 \ldots\}, T_{r_j} \subseteq T$, and hierarchically clustering tags in global interest areas (*interests tree).*

2. *Mapping.* Mapping of actor interest profiles (ATPs) and resource properties (RTPs) to the created tree, to construct tagging profiles.

3. *Predicting.* Calculating similarities of ATPs and RTPs, and applying trust prediction to determine potential trust relations.

7.4.1 Hierarchical Clustering of Global Interest Areas

The advantage of clustering related tags is twofold: (i) we are able to identify widely used synonyms and equal, but differently written, tags (including singular/plural forms), and (ii) we are able to identify tags with similar meanings or tags mostly used in combination. To this end, we build from the captured tagging actions a global interests tree by applying hierarchical clustering. This interests tree reflects which tags are generally applied to resources in combination, and therefore, their relatedness.

The utilized concepts are well-known from the area of information retrieval (see for instance [103]), however, while they are normally used to determine the similarities of documents based on given terms, we apply them in the opposite way. This means we determine term, i.e., tag, similarities based on given tag sets (profiles that are interpreted as kinds of documents).

The tag frequency vector $\mathbf{t_x}$ (Equation 7.1) describes the frequencies f the resources $R = \{r_1, r_2 \dots r_j\}$ are tagged with tag $t_x \in T$ globally, i.e., by all actors N.

$$\mathbf{t_x} = \langle f(r_1), f(r_2) \dots f(r_j) \rangle \tag{7.1}$$

The tag frequency matrix tfm (7.2), built from tag frequency vectors, describes the frequencies the resources R are tagged with tags $T = \{t_1, t_2 \dots t_x\}$.

$$tfm = \langle \mathbf{t_1}, \mathbf{t_2} \dots \mathbf{t_x} \rangle_{|R| \times |T|} \tag{7.2}$$

The popular $tf^{*}idf$ model [103] introduces tag weighting based on the relative distinctiveness of tags (Equation 7.3). Each entry $tf(t_x, r_j)$ in tfm is weighted by the log of the total number of resources $|R|$, divided by the amount $n_{t_x} = |\{r_j \in R \mid tf(t_x, r_j) > 0\}|$ of resources the tag t_x has been applied to.

$$tf^{*}idf(t_x, r_j) = tf(t_x, r_j) \cdot \log \frac{|R|}{n_{t_x}} \tag{7.3}$$

Finally, the cosine similarity, a popular measure to determine the similarity of two vectors in a vector space model, is applied (Equation 7.4).

$$\mathrm{sim}(\mathbf{t_x}, \mathbf{t_y}) = \cos(\mathbf{t_x}, \mathbf{t_y}) = \frac{\mathbf{t_x} \cdot \mathbf{t_y}}{||\mathbf{t_x}|| \cdot ||\mathbf{t_y}||} \tag{7.4}$$

We perform hierarchical clustering to the available tag vectors. This clustering approach starts by putting each tag vector $\mathbf{t_x}$ into a single cluster, and compares cluster similarities successively. Tag clusters are then merged bottom-up when the similarity measurement result exceeds predefined thresholds. The output of clustering is a hierarchical tree structure, i.e., a dendrogram, reflecting global interest areas and their similarity (Figure 7.4). The details of the algorithm are shown in Section 7.5.

The approach can be further refined by applying the concept of latent semantic indexing (LSI) [23]. However very common in information retrieval, this method demands for carefully selected configuration parameters not to distort the similarity measurement in our case. Our approach applies hierarchical clustering, which means tag clusters are merged

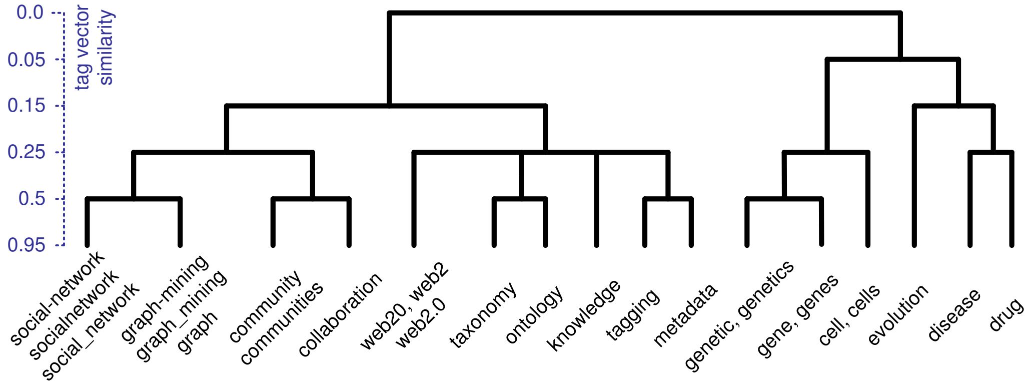


Figure 7.4: A small part of the `citeulike` global interests tree.

based on varying similarity thresholds. Thus, we do not necessarily need a further level of fuzziness introduced by LSI.

7.4.2 Tagging Profile Creation

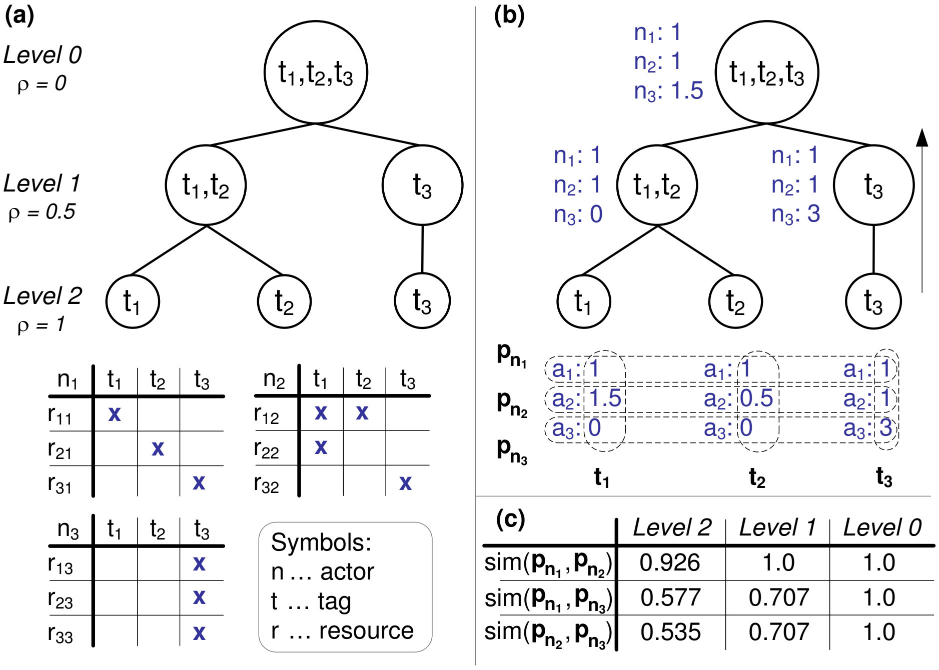
As mentioned earlier, we create tagging profiles for both actors and resources. While ATPs describe the interests of actors, RTPs reflect features and properties of resources. The performed steps to create either kind of tagging profiles are almost identical. Therefore we show exemplarily the construction of ATPs in Figure 7.5. For RTPs the transposed tagging matrices are used.

The upper part of the left picture (Figure 7.5(a)) depicts the tree of global interests, created in the previous step. The lower part describes tagging matrices of three actors, e.g., actor n_1 tags resource r_{11} with tag t_1. In Figure 7.5(b), these tagging activities are weighted and mapped to the *bottom* clusters of the interests tree (here: level 2). For this purpose, the impact w of each tag t_x on n_i's ATP is calculated according to Equation 7.5, assuming that the sum runs over all resources $R_{n_i} \subseteq R$ that are tagged by n_i with tag $t_x \in T_{r_j}$. Therefore, the more tags are assigned to one resource $r_j \in R_{n_i}$, the less impact one tag t_x has on the description of the resource. The assigned weights to each cluster build the ATP vectors $\mathbf{p_{n_i}}$ (see Figure 7.5(b)).

$$w(n_i, t_x) = \sum_{\forall r_j \in R_{n_i}} \frac{1}{|T_{r_j}|} \tag{7.5}$$

In the next steps the ATP vectors are aggregated and propagated to the upper levels, by simply building the average of all weights that are assigned to child clusters. Hence, new ATP vectors on a higher and more abstract level are built. Finally, the root of the interests tree is reached according to Figure 7.5(b).

For each actor either all tagged resources or representative subsets (e.g., the most frequently tagged resources) are used to create the ATP. Such a general ATP reflects an actor's general interests. The same can be applied to resources, where RTPs describe their general use. Instead, tailored ATPs reflect an actor's understanding and apprehension of a particu-



n_1	t_1	t_2	t_3
r_{11}	x		
r_{21}		x	
r_{31}			x

n_2	t_1	t_2	t_3
r_{12}	x	x	
r_{22}	x		
r_{32}			x

n_3	t_1	t_2	t_3
r_{13}			x
r_{23}			x
r_{33}			x

Symbols:
n … actor
t … tag
r … resource

	Level 2	*Level 1*	*Level 0*
sim($\mathbf{p}_{n_1}$,$\mathbf{p}_{n_2}$)	0.926	1.0	1.0
sim($\mathbf{p}_{n_1}$,$\mathbf{p}_{n_3}$)	0.577	0.707	1.0
sim($\mathbf{p}_{n_2}$,$\mathbf{p}_{n_3}$)	0.535	0.707	1.0

Figure 7.5: An example for tag mapping and ATP comparison: (a) interest tree and actor tagging actions. (b) creating ATPs by mapping tagging actions to the tree. (c) calculating ATP similarities on different tree levels.

lar and carefully selected subset of resources. For instance, in the case of trust prediction in a collaborative environment, resources might be selected according to their importance in an ongoing task. According to Figure 7.5, this means each actor tags exactly the same resources, i.e., $r_{x1} = r_{x2} = r_{x3}\, \forall x \in \{1,2,3\}$. On the other hand, tailored RTPs can be used for trustworthy replacements of one resource with another one, on which a particular subset of actors have similar views.

7.4.3 Trust Prediction

The profile similarity of two actors n_i and n_j is determined by the cosine of the angle between their ATP vectors $\mathbf{p_{n_i}}$ and $\mathbf{p_{n_j}}$ (cosine similarity). This similarity can be calculated for each level of the global interests tree, whereas the similarity increases when walking from the bottom level to the top level. Figure 7.5(c) shows the similarities of ATP vectors on different levels for the given example.

However, the higher the $level$ and the more tags are included in the same clusters, the more fuzzy is the distinction of tag usage and therefore the similarity measurement. Thus, we introduce the notion of reliability ρ (Equation 7.6) of a tagging profile similarity measurement in a tree with $numLevels$ in total.

$$\rho(\text{sim}(n_i, n_j)) = \frac{level}{numLevels} \tag{7.6}$$

For mirrored trust τ_M (Equation 7.7), as defined in Section 7.2, only profile similarities and their reliability are used to predict a level of potential trust.

$$\tau_M(n_i, n_j) = \text{sim}(n_i, n_j) \cdot \rho(\text{sim}(n_i, n_j)) \tag{7.7}$$

Teleported trust τ_T (Equation 7.8) means that an existing directed trust relation $\tau(n_i, n_k)$ from actor n_i to n_k is teleported to a third actor n_j depending on the similarity of n_k and n_j. This teleportation operation $\otimes$ can be realized arithmetically or rule-based.

$$\tau_T(n_i, n_j) = \tau(n_i, n_k) \otimes (\text{sim}(n_k, n_j) \cdot \rho(\text{sim}(n_k, n_j))) \tag{7.8}$$

7.5 Implementation

In this section we introduce the architectural components of our trust bootstrapping and prediction framework. Our architecture has been implemented on top of Web service technology suitable for distributed, large-scale environments. Furthermore, we detail the clustering algorithm by showing the steps needed to create hierarchical, tag-based interests trees.

7.5.1 Reference Architecture

The presented architecture evolved from our previous efforts in the area of trust management in service-oriented systems (see Chapter 3 and 4 for details on the VieTE framework).

Our architecture consists of the following main building blocks:

- *Tagging and Social Network Web Services* facilitate the integration of existing systems and the usage of external data sources. Tagging and social networks, for example, interaction graphs, can be imported via Web services.
- *Data Provisioning* comprises a set of `Providers`. We separated these providers in resource-centric (e.g., *Tag*, *Resource*, *Actor*) and trust-centric blocks. Providers enable access to *Tagging Data* and *Social Network Data* using the messaging system JMS[2]. We use the WS-Resource Catalog (WS-RC) specification[3] to manage resources in the system.
- *Trust Prediction* components consist of `Management` support, responsible for the ATP/RTP creation and tailoring of tagging profiles, and `Measurement` support used for various algorithmic tasks such as trust prediction and similarity calculation.
- *Trust Prediction Web Service* enables access to predicted trust in a standardized manner. We currently support SOAP-based services but our system can be easily enhanced by adding RESTful services support.

[2] `http://java.sun.com/products/jms/`

[3] `http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/`

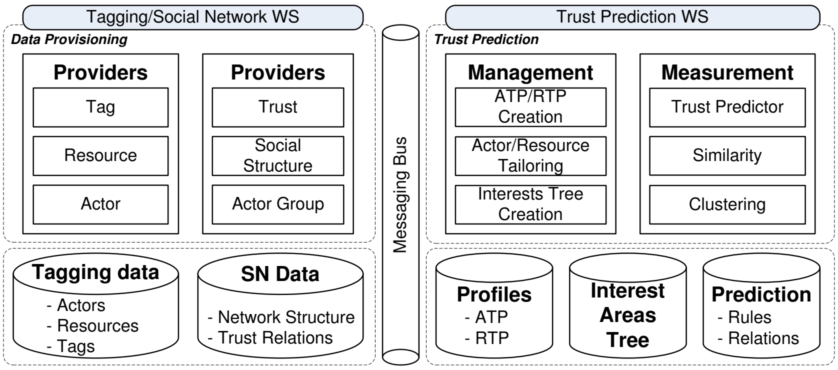


Figure 7.6: Reference architecture enabling trust prediction in social network platforms.

7.5.2 Hierarchical Clustering Algorithm for Interests Tree Creation

We detail our clustering approach to interests tree creation as illustrated by Algorithm 7. Briefly, the clustering starts by putting each tag vector $\mathbf{t_x}$ (see Equation 7.1 in Section 7.4)

Algorithm 7 Hierarchical clustering of global interest areas

```
/* create tag frequency matrix */
$\langle N, R, T\rangle = retrieveTaggingDataFromDB()$
$tfm = \emptyset$
for each $t_x \in T$ do
    $\mathbf{t_x} = createTagFrequencyVector(t_x, \langle N, R, T\rangle)$
    $addToTagFrequencyMatrix(tfm, \mathbf{t_x})$
/* weight single tag entries */
for each $t_x \in T$ do
    for each $r_j \in R$ do
        $tf(t_x, r_j) = extractValue(tfm, t_x, r_j)$
        $updateValue(tfm, tf(t_x, r_j) * idf(t_x, r_j))$
/* perform hierarchical clustering */
$\vartheta[\,] = \{0.95, 0.5, 0.25, 0.15, 0.05, 0.0\}$
$Cluster[\,][1] = createClusterForEachTag(tfm)$
for $i = 1 \rightarrow |\vartheta[\,]|$ do
    for $u = 1 \rightarrow |Cluster[\,][i]|$ do
        $C_u = Cluster[u][i]$
        if $\neg isMerged(C_u)$ then
            $C_{sim}[\,] = \{C_u\}$
            for $v = u + 1 \rightarrow |Cluster[\,][i]|$ do
                $C_v = Cluster[v][i]$
                if $\neg isMerged(C_v)$ and $getSimilarity(C_u, C_v) \geq \vartheta[i]$ then
                    $addToClusterArray(C_{sim}[\,], C_v)$
            $C_m = mergeClusters(C_{sim}[\,])$
            $addToClusterArray(Cluster[\,][i+1], C_m)$
```

into a single cluster, and comparing cluster similarities successively. After comparing each cluster with each other, all clusters having cosine similarities above a predefined threshold ϑ and have not been merged yet, are combined to single clusters. Then, ϑ is lowered and the algorithm compares again all available clusters. Finally, all tag vectors are merged in one single cluster, resulting in a tree structure, that reflects the global interests (Figure 7.4 in Section 7.4). The function $getSimilarity()$ implements an average similarity measurement by comparing artificial tag vectors that are based on the averages of all vectors within respective clusters.

7.6 Evaluation and Discussion

We evaluate and discuss our new tagging profile creation and similarity measurement approach using real-world data sets from the popular `citeulike`[4] community. `Citeulike` is a platform where users can register and tag scientific articles. But before we used this tagging data, we performed two refactoring operations: (i) removing tags reflecting so-called stop words, e.g., `of`, `the`, `in`, `on` etc., resulting from word groups which are sometimes separately saved; (ii) filtering of tags reflecting ambiguous high level concepts such as `system`, `paper`, `article`; (iii) deleting tags not related to the features or properties of resources, including `toread`, `bibtex-import`, `important`. These steps reduce the available 'who-tagged-what' data entries from 5.1 million to 4.4 million.

7.6.1 Interests Tree Creation

For the later following ATP and RTP creation, all actor or resource tags are mapped to the same global interests tree. Therefore, the tree must be broad enough to contain and cover the most common tags. Due to the huge size of the data set, we picked the 100 articles to which most distinct tags have been assigned, and use all tags which have been applied to at least five of these articles.

In `citeulike` users are free to add arbitrary self-defined tags, raising the problem of differently written tags reflecting the same means. For instance the tag `social-network` appears written as `socialnetwork`, `social_networks`, `social-networks` etc., all meaning the same. To realize their equality, we start by clustering tags with a comparably high similarity (≥ 0.95), and consider these clusters as our initial cluster set. As long as differently written, but equally meant tags are used with a similar frequency and distribution among resources, we can capture their potential equality, otherwise the impact of alternative tags is comparably low and negligible. Then, we compare tag clusters applying much lower similarity thresholds (≤ 0.50) to capture tags reflecting similar concepts.

Table 7.1 summarizes the tagging data properties used to construct the interests tree. This tree consists of six levels, starting with 760 clusters on the lowest one (see Figure 7.4 in Section 7.4). The utilized algorithm is detailed in the next section.

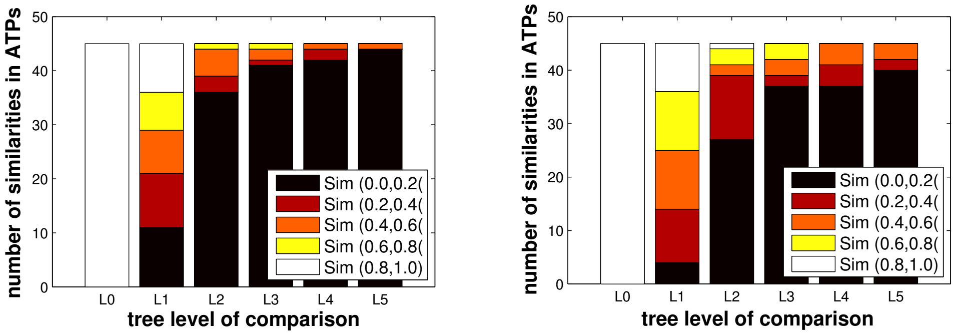
[4] `http://www.citeulike.org/faq/data.adp`

Metric	Filtered data set	Interests tree
Num. of articles	1020622	100
Num. of articles recognized by more than 50 users	25	21
Num. of distinct tags	287401	760
Num. of distinct tags applied by more than 500 users	272	-
Num. of distinct users	32449	-
Average num. of tags per article	1.2	157
Average num. of users per article	3.5	37

Table 7.1: Data properties for constructing the global interests tree.

7.6.2 Profile Mapping and Trust Prediction

We determine (i) for 10 highly active users the similarities of their general ATPs, and (ii) for 10 users in the area of the *social web* their tailored ATPs. For the first test we selected the 10 articles which have been tagged with most distinct tags. Then, for each of these articles, we picked the user who applied most tags to it. Therefore, we get users, who tagged highly recognized but different articles. We create the ATPs by retrieving all tags that each of the selected users applied to his/her 10 most tagged articles (between 50 and 300 tags per ATP). We compare all ATPs with each other (in total 45 comparisons) on each level of the interests tree. The results are depicted in Figure 7.7(a). As expected, level 5 comparisons result mostly in no similarity, only two ATPs have a similarity of 0.42 on this level. The amount of similar ATPs in different similarity classes increases when we compare them on higher levels of the interests tree. On level 0, of course, all ATPs are equal, because all tags are merged in the same cluster. These results show that our approach of indirect similarity measurement provides distinguishable similarity results on different levels of the tree.



(a) General ATP similarity.

(b) Tailored ATP similarity.

Figure 7.7: ATP similarity in `citeulike` on different levels.

In a second experiment we measure similarities of tailored ATPs. We restrict the tags used for ATP creation to a subset of resources, and consider only tags assigned to articles in the field of the *social web*. We filter all articles, which are not linked to the `citeulike` groups *Semantic-Social-Networks*[5], *social_navigation*[6], and *Social Web*[7]. The ATP similarity results for the 10 most active users spanning these groups are depicted in Figure 7.7(b). Obviously, due to the restricted tag set and a common understanding of tag usage, ATP similarities, especially on level 2 to 4, are significantly higher than in the general comparison before. Furthermore, we compare two sets of users, interested in computer science, but only members of one set participate in *social web* groups. Their general ATPs are largely similar on level 1 to 3, because all users assigned many general tags related to computer science. However, if we compare both groups' ATPs tailored to the *social web*, there is nearly no remarkable similarity until level 1. We conclude, that tailored profiles are key to more precise trust prediction.

[5] `http://www.citeulike.org/groupfunc/328/home` (82 users, 694 articles)
[6] `http://www.citeulike.org/groupfunc/1252/home` (20 users, 507 articles)
[7] `http://www.citeulike.org/groupfunc/3764/home` (27 users, 444 articles)

Part III

Trust-based Service-centric Applications

CHAPTER 8

Context-aware Interaction Models in Virtual Organizations

Outline. We go one step further towards the realization of the Expert Web use case (as discussed in Chapter 5), and introduce fundamental concepts to allow flexible expert discovery and involvement in process-oriented environments.

Contents

8.1 Motivation

Recently, supporting the flexible creation of virtual organizations (VOs), spanning companies, communities, and individuals, to compete with global enterprises has gained major research attention. A virtual organization is a *temporary alliance of enterprises that come together to share skills or core competencies and resources in order to better respond to business opportunities, and whose cooperation is supported by computer networks* [17].

A process in a VO spans multiple organizations, whereas each task is either performed by only one physical company or processed by various partners. While the interfaces and flow of information and goods between the single task owners are pre-planned, human interactions are usually not static. Especially in those cases where processes have not been executed several times; thereby providing historical information, need dynamic interactions of participants to adapt and optimize workflows, or to solve unforeseen problems. In such scenarios we distinguish between two fundamental kinds of human interactions: (i) organization-internal interactions, such as the communication and coordination between members of the same company; and (ii) cross-organizational interactions that take place between different physical companies.

Typical research challenges that arise in such scenarios [116] deal with the discovery of people in partner organizations, accounting for contextual constraints (online presence state, contact details, preferred communication channels), personal profiles (skills, certificates, collected experiences), and personal relations based on previous joint collaborations in similar situations. Figure 8.1 depicts a typical scenario, where the task "Mechanical Specification" of a construction process is jointly performed by the customer organization (that participates in the VO itself), and a construction office.

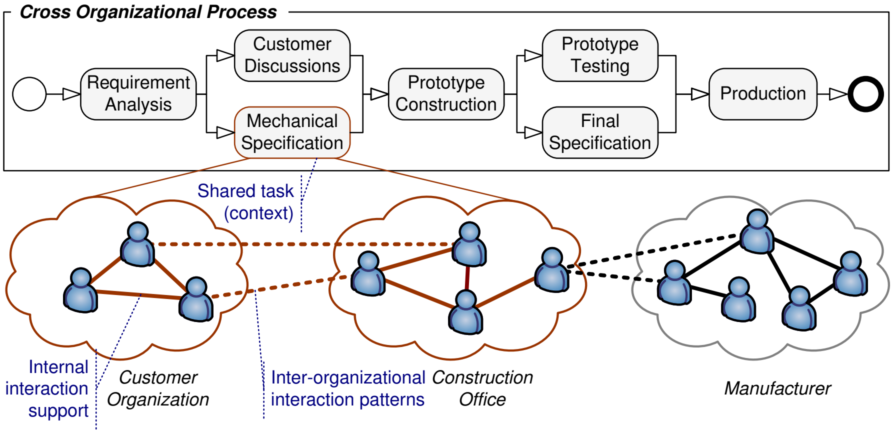


Figure 8.1: Human interactions in processes spanning organizations.

Consider the following scenario in Figure 8.1: The construction office creates the mechanical specification for a part required by a customer. However, there are various open issues regarding the efficient production later on. Therefore, an immediate discussion between the customer who has certain requirements, the construction office that designs the prototype, and the manufacturer who is able to optimize the manufacturing process, is needed. Fundamental problems in this scenario include: Who are the persons with the required knowledge in the respective organizations? How can they be contacted, quickly informed about the problem, and involved in discussions? Who are distinguished third party experts that could assist to come to an agreement? What persons can be trusted to make the right decisions as they may have dealt with similar situations before?

This chapter comprises the following contributions:

- *Context-aware Human Interaction Support.* We highlight novel concepts for context-aware human interaction support in process-oriented collaborations. Therefore, we introduce the COIN[1] architecture, basic technologies for realizing cross-organizational processes, as well as fundamental human interaction concepts.

- *Expert Ranking Approach.* We deal with an expert ranking approach based on interaction behavior and trust, and demonstrate the integration of our work in a virtual production planning environment.

[1] EU FP7-216256 project 'COIN': `http://www.coin-ip.eu`

8.2 Foundational Concepts

Before we deal with our approach to flexible human involvement in cross-organizational processes, we outline the COIN project that represents the underlying basis for our work.

8.2.1 The COIN Enterprise Collaboration Architecture

The COIN project aims at providing an open, self-adaptive integrative solution for *Enterprise Interoperability* and *Enterprise Collaboration*. Service orientation is a well-suited and widely adopted concept in collaboration scenarios, therefore, COIN utilizes state of the art SOA concepts, including Semantic Web Technologies and Software-as-a-Service (SaaS) models (see [39] for more details). With respect to Enterprise Collaboration, COIN supports numerous features that focus on product development, production planning and manufacturing, and project management in networks of enterprises. As a fundamental aspect, human interactions exist in all forms and phases of virtual organizations and play a major role in the success of collaborations within enterprise networks. Therefore, understanding human interactions and providing advanced support for efficient and effective interactions, is one of the key objectives in COIN's Enterprise Collaboration research track.

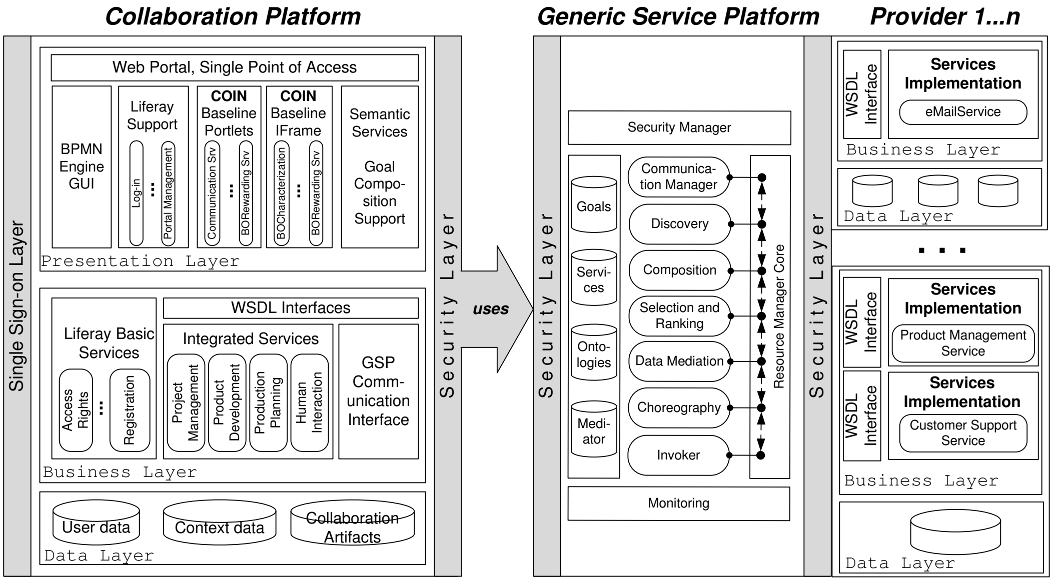


Figure 8.2: The COIN Framework enabling cross-organizational collaborations.

The COIN Framework (see Figure 8.2) consists of (i) the Collaboration Platform (CP) that provides fundamental features that are required in (nearly) every collaboration scenario, and (ii) the Generic Service Platform (GSP) that allows extensions with services following the SaaS model from third party providers. The CP is designed for and tightly coupled to a Liferay[2] community portal that provides an effective way to configure and per-

[2] Open Source Portal Liferay: `http://www.liferay.com`

sonalize the CP for specific end-users by providing customized services and tools. Single sign-on- and security mechanisms span services and tools across layers. The GSP relies on semantic web technologies, implemented by the WSMX[3] environment and is utilized to discover, bind, compose, and use third-party services at run time.

Because of its extensibility and configurability, the COIN platform can be applied in a wide variety of different collaboration scenarios, ranging from traditional production planning to social campaigning and interest group formations in professional virtual communities. For enabling context-aware interactions, the following baseline components are of major interest (i) user data, including skills and interest profiles, (ii) context data, such as current ongoing activities and user preferences, (iii) integrated baseline services for communication and coordination (e.g., e-mail notifications, and instant messengers), (iv) the GSP as the platform to host extended human interaction services.

8.2.2 Process Models for Virtual Organizations

COIN collaborative *Production Planning Services* realize innovative solutions in the field of production planning. The C3P (*Collaborative Production Planning Platform*) service is a graphic environment focusing on collaborative creation of production processes. Companies can conveniently plug themselves to the system, invite potential partners and contribute to the definition of the entire production plan. Furthermore, they collaboratively solve arising problems during the execution using human interaction services. The flow of steps to manufacture a product is defined on two different workflow levels:

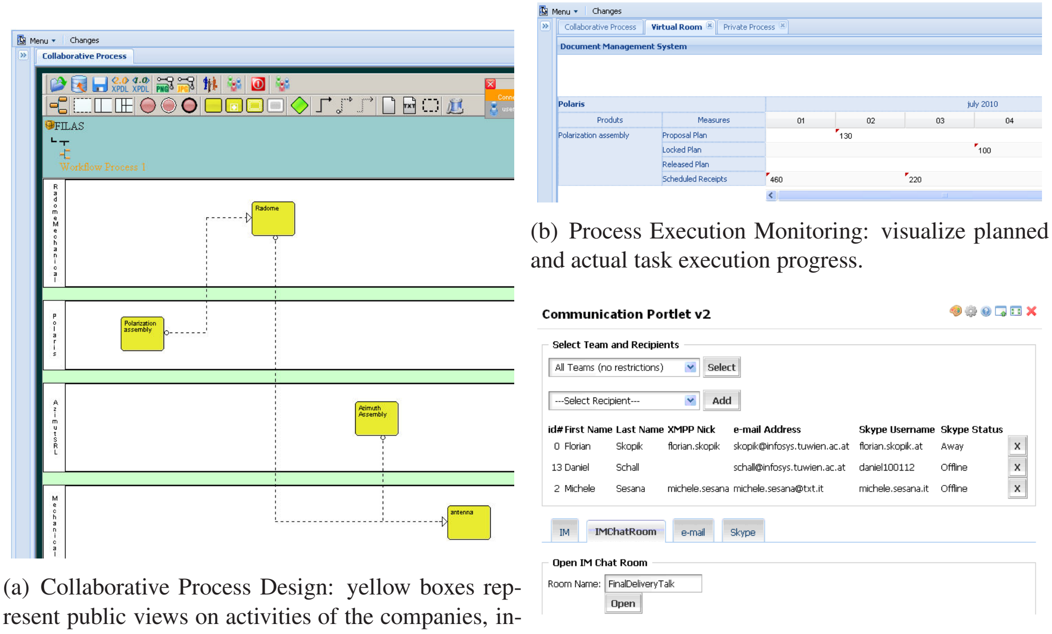
- *Collaborative Public Processes* are defined as XPDL[4] workflows; whereas each step has at least one responsible partner assigned. The process steps are defined collaboratively by partners, and represent interfaces that hide company-internal (sub-)processes. The *Collaborative Public Process Management Service* allows multiple users to modify the same process concurrently, for instance, inserting new activities, splitting activities in different steps provided by different partners, deleting activities and defining relations between steps.

- *Private Processes* define workflows on company-level. Each company describes and imports its own private (sub-)processes involving company resources, such as personnel, material, and machines. Starting from a step of the public process this module allows a particular company to connect its private processes to address the goal of the related public process. Because of privacy issues and protecting know-how, private processes are available to persons of the owning company only.

The meeting points between different partners participating in the collaborative public process are virtual rooms that are linked to the arrows of the workflow model (see Figure 8.3(a) and 8.3(b)). Actors can collaboratively define interfaces between process steps, e.g., regarding the shipment of goods. Furthermore, they solve arising problems to find a final agreement of the production plan. Figure 8.3(a) depicts a public process. Following the

[3] WSMX: Web Services Modelling eXecution Environment: `http://www.wsmx.org`

[4] XPDL: XML Process Definition Language: `http://www.wfmc.org/xpdl.html`



Polaris		july 2010			
Produts	Measures	01	02	03	04
Polarization assembly	Proposal Plan		130		
	Locked Plan				100
	Released Plan				
	Scheduled Receipts	460		220	

Communication Portlet v2

Select Team and Recipients

All Teams (no restrictions) Select

---Select Recipient--- Add

id#	First Name	Last Name	XMPP Nick	e-mail Address	Skype Username	Skype Status	
0	Florian	Skopik	florian.skopik	skopik@infosys.tuwien.ac.at	florian.skopik.at	Away	X
13	Daniel	Schall		schall@infosys.tuwien.ac.at	daniel100112	Offline	X
2	Michele	Sesana	michele.sesana	michele.sesana@txt.it	michele.sesana.it	Offline	X

IM | IMChatRoom | e-mail | Skype

Open IM Chat Room

Room Name: FinalDeliveryTalk

Open

(a) Collaborative Process Design: yellow boxes represent public views on activities of the companies, including items manufacturing or testing; in some cases goods shipping are included in the item manufacturing operation.

(b) Process Execution Monitoring: visualize planned and actual task execution progress.

(c) Context-aware Expert Involvement: contact the best experts in their fields considering contextual constraints including communication channels and online presence.

Figure 8.3: Flexible expert involvement in running processes.

BPMN[5] standard the horizontal white areas represent the customers, the OEM (first tier of the supply chain) and the different suppliers (second supply chain tier). The yellow boxes reflect the public view on the steps of the chain to complete the final item manufacturing. Temporal dependencies of activities are clearly visible through the arrows linking the public activities that compose the public process. For the sake of simplicity not all activities are represented in this example.

This section discussed the conceptual and technical grounding of process-oriented environments as used in the previously introduced Expert Web use case (see Chapter 5). However, in this thesis we focus on flexible collaboration environments and dynamically changing social relations and member expertise. Therefore, we further discuss in this chapter the role of human interactionsm their mining, and dynamic profile updates (such as experience and skills) for supporting fundamental expert discovery mechanisms.

8.2.3 Human-Interaction Support in Virtual Organizations

Virtual Organizations pose additional challenges to human interaction support. VOs are typically temporary alliances that form and dissolve again. Various actors are involved in such VOs collaborating and working on joint activities. However, finding the right person to work on tasks or to solve emerging problems is challenging due to scale (number of

[5] BPMN: Business Process Modeling Notation: `http://www.bpmn.org`

people involved in VOs) and the temporary nature of formations. Furthermore, actor skills and competencies evolve over time requiring dynamic approaches for the management of these actor properties. In this work, we propose context-aware techniques and tools to address fundamental issues in such collaboration environments: how to find the right person and collaborate with that person using the best suitable communication and interaction channel? We propose a combination of the following concepts (as separately discussed in previous chapters) to address the need for context-aware interactions in VOs:

- Mining of interactions to determine patterns, for example delegation patterns, user preferences, and user behavior (described by multiple metrics).
- Managing context information to select suitable interaction and communication channels.
- Trust inference methods based on social relations to influence communication patterns [116].

Furthermore, human interactions need to be supported in service-oriented systems. Using traditional communication tools (e.g., e-mail, instant messaging tools, or Skype) only may not be well suited for that purpose, especially when neglecting context. To address human interactions in SOA, we propose Human-Provided Services [106] that can be additionally utilized for providing 'trusted online help and support' enabling experts to define services based on their skills. This approach makes the flexible involvement in workflows possible without designing such interactions a-priori. A set of tools and services support human interactions including: (i) *Communication services*: chat, e-mail, IM, Skype, and various notification services (ii) *Activity management service* managing flexible collaboration structures and artifacts used in collaborations (documents, resources, etc.); (iii) *Profile management service* for storing user skills and capabilities

Specifically, context is used in various ways to support adaptive communication and collaboration: (i) Communication channels can be pre-selected based on user presence (online status), privacy rules, and urgency (e.g., e-mail v.s. VoIP v.s. SMS). (ii) Users are assigned to activities based on their skills but also social preferences of other users working on joint activities. (iii) Expert finding based on reputation in a certain field, for example, with respect to activities that need to be performed.

In the rest of this chapter, we focus exemplarily on the latter, i.e., expert discovery and ranking.

8.3 Context-aware Human Interaction Support

We highlight an expert query and ranking approach and demonstrate its application in the process-oriented cross-organizational collaboration environments.

8.3.1 Context Model

Observing and mining interactions enables the calculation of metrics that describe the collaboration behavior of network members regarding certain activities. Furthermore, mining social network data determines reputation of actors and their trust relations (such as in friend networks). The utilized context model, as presented in Chapter 3, centers all data about actors, profiles, relations, and resources, around activities. Furthermore, metrics that describe actor behavior with respect to different activity types are part of this model. These data are the basis for ranking and selecting experts.

8.3.2 Applications

We outline flexible expert involvement and management of communication among two different companies in a shared workflow. The following software modules are used for flexible human interactions in processes: (i) *COIN Baseline* including a central database to store and manage profiles of individuals, teams and organizations; (ii) *Activity and Task Models* that are used to infer the context of ongoing work. This information improves the expert search by accounting for experience and expertise; (iii) *C3P Production Planning Software*, utilizing concepts of public and private workflows presented before; (iv) *Communication Services* to actually involve experts via e-mail, instant messaging, or Skype.

Figure 8.3 depicts the single steps of involving experts. In (a), still in the planning phase, partners can be involved to discuss the planned process, while in (b) the actually executed state and emerging problems are discussed. For that purpose, contextual information is derived from a task's execution, including its type, temporal constraints, and the owning company, to discover assistance. This means, the requester for an expert, i.e., the activity owner, can specify an expert search query according to external constraints; for instance, urgent support needs an expert to be currently online and highly responsive; or tasks carrying company sensitive information should not be shared with external people.

8.3.3 Expert Ranking and Query Mechanisms

We rank members N in the Web of Trust $G = (N, E)$ and determine experts with the *Promethee* approach [14] based on multiple criteria, obtained from mining interactions as mentioned above. Our overall approach to determine the best available expert(s) on request is depicted in Algorithm 8.

Algorithm 8 Context-aware trusted expert discovery based on multiple criteria.

```
Input: search query
Compute:
  1. Filter experts according to mandatory constraints.
  2. Select ranking criteria and order.
  3. Assign weights to criteria.
  4. Rank remaining experts.
  5. Pick on or more of the top ranked experts.
Output: best available expert
```

First (1), all experts that do not fulfill certain constraints, mandatory to support a given activity, e.g., online state, company membership, are sorted out. Afterwards (2), the activity leader sets ranking criteria and their order, for instance $experience \succ reputation \succ responsiveness$. The order influences the importance (weights) of criteria (3). Then the actual ranking is performed (4), and from the resulting list experts are manually picked (5).

We denote $P_j(n_1, n_2) \in [0, 1]$ as the priority of the choice of node n_1 over n_2 in G with respect to criteria j. For instance, expert n_1 is preferred over n_2 regarding metric $experience$. Since we rank experts with respect to multiple criteria, i.e., values of k metrics, we aggregate priorities as shown in Equation 8.1. The weight w_j of each criterion j is derived from the specified order of important metrics in the search query.

$$\pi(n_1, n_2) = \sum_{j=1}^{k} P_j(n_1, n_2) w_j \tag{8.1}$$

Outrankings (Equation 8.2, 8.3) compare a choice of an expert n_1 with the $|N|-1$ other choices in the set of all available experts N. The positive outrank Φ^+ determines how n_1 is outranking all other experts, and Φ^- determines how all other experts are outranking n_1. The higher the value of Φ^+, and the lower the value of Φ^-, the better is the choice n_1.

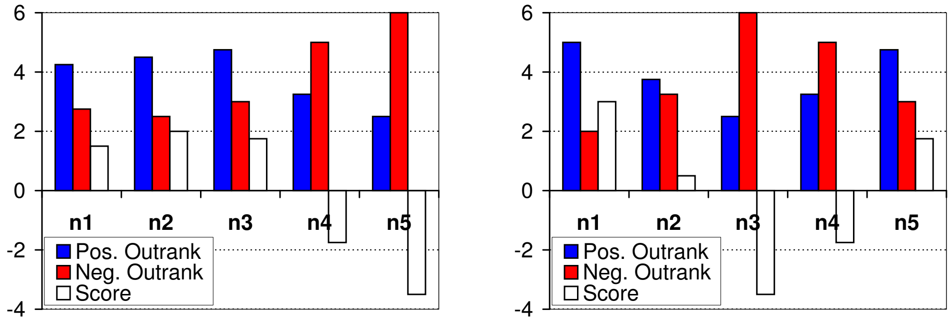
$$\Phi^+(n_1) = \frac{1}{|N| - 1} \sum_{n_x \in N} \pi(n_1, n_x) \tag{8.2}$$

$$\Phi^-(n_1) = \frac{1}{|N| - 1} \sum_{n_x \in N} \pi(n_x, n_1) \tag{8.3}$$

Finally, the score of an expert is calculated by Eq. (8.4).

$$\Phi(n_1) = \Phi^+(n_1) - \Phi^-(n_1) \tag{8.4}$$

We demonstrate the application of the described *Promethee* approach [14] with a short example. Assume we rank experts according to different metrics $experience$, $reputation$, $responsiveness$ ($\in$ [0,100]) with two different queries $Q_1 = \{exp \succ rep \succ resp\}$ and $Q_2 = \{resp \succ rep \succ exp\}$. Figure 8.4 compares expert ranks, and Table 8.1 shows the detailed ranking results. Note, the impact of k metrics vary with their *pos*ition in the queries, and weights are defined as $w_j = 2^{k-pos(j)}$.



(a) $Q_1 = \{exp \succ rep \succ resp\}$.

(b) $Q_2 = \{resp \succ rep \succ exp\}$.

Figure 8.4: Expert ranking results for Q_1 and Q_2.

Complexity of the *Promethee* approach including speedup methods and parallel computation is discussed in [26]. The method requires $\mathcal{O}(kn^2)$ independent operations to build the outrankings. The number of operations grows very large as the number of criteria (k) and alternatives (n) increases. However, the method can be optimized by parallel computation of operations [26].

expert	experience	reputation	responsiveness	Φ_{Q_1} *(rank)*	Φ_{Q_2} *(rank)*
n_1	50	50	50	1.5 *(r3)*	3 *(r1)*
n_2	75	25	25	2 *(r1)*	0.5 *(r3)*
n_3	100	0	0	1.75 *(r2)*	-3.5 *(r5)*
n_4	0	100	0	-1.75 *(r4)*	-1.75 *(r4)*
n_5	0	0	100	-3.5 *(r5)*	1.75 *(r2)*

Table 8.1: Example rankings accounting for experience, reputation, and responsiveness.

Example Scenario. A manufacturer from China and an assembler from Italy work together on the assembly of a product. The manufacturer in China has to send goods to the company in Italy. Unforeseen problems may happen at China's customs when exporting certain parts. In this case persons from both companies can collaborate in the virtual room (see Figure 8.3(b)), sharing the current and the adapted production plan, uploading documents from China's custom office, chatting or talking via Skype to find a solution. When the required set of skills, such as far-east custom policies expertise, are not satisfied, third-party experts from outside the currently executed process can be involved through an expert search query. The discussion participants in the virtual room can decide about useful contextual constraints and discover a set of people who match the search query. Finally, the expert requester(s) may pick one or more people to be contacted (visualized in Figure 8.3(c)).

8.4 Results and Findings

After extensive discussions with COIN end-user partners, such as Poyry[6], the system is applied in their business cases. The following results can be mentioned: (i) *Enhanced expert discovery mechanisms.* By considering not only static competencies, such as official certificates and education, but also dynamically changing experiences, experts can be selected more precisely; especially when accounting for particular contextual constraints, such as online presence for immediate responses or organizational memberships. (ii) *Significantly reduced response times.* By automatically selecting preferred communication channels, experts can be faster involved in ongoing collaborations. Communication channels are selected based on working time, location of people, and their current activities (all information obtained from the context model). (iii) *Harnessing distributed expertise.* Involving experts from various physical companies in the same virtual organization massively extends the pool of available skilled persons who can assist in ongoing collaborations.

[6] Pöyry Group: `http://www.poyry.com`

Besides these positive aspects, we will conduct further research to deal with negative side effects, such as (i) *Privacy concerns* due to monitoring and mining interactions (also see Chapter 11 on privacy issues), (ii) *Complex adaptations and extensions of the context model* to suitably reflect the real environment.

CHAPTER 9

Trust-based Discovery and Interactions in Expert Networks

Outline. We deal with an advanced expert discovery approach in the previosuly introduced Expert Web. For that purpose, we extend the popular HITS algorithm[65] with context-sensitive personalized trust weightings.

Contents

9.1 Motivation

Web services have pioneered the way for a new blend of composable systems. Services already play an important role in fulfilling organizations' business objectives because process stakeholders can design, implement, and execute business processes using Web services and languages such as the Business Process Execution Language (BPEL) [92]. A broad range of services is increasingly found in open Web-based platforms. Users and developers have the ability to use services in various applications because services offer well-defined, programmable, interfaces.

In process-centric collaboration, a top-down approach is typically taken by defining process activities and tasks prior to deploying and executing the process. Before creating the model, the designer must fully understand each step in the process. Flexibility in such composition models is limited since unexpected changes require remodeling of the process. Such changes may cause exceptions, disrupting the normal execution of the process. It is important to support *adaptivity* in collaborations and compositions. An important role towards adaptive processes plays the ability to support the execution of ad-hoc activities and flexibility in human interactions to react to unexpected events. While the process-centric collaboration approach follows a top-down methodology in modeling flows, ad-hoc flows in flexible collaborations emerge at run-time. A run-time environment constraints the execution of flows. Such constraints are, for example, the availability of resources, services, and people.

In this chapter, we focus again on the Expert Web use case, based on the concept of Human-Provided Services (HPS) [106]. We discuss the discovery and interactions in mixed service oriented systems comprising HPS and software-based services (SBS). Experts offer their skills and capabilities as HPS that can be requested on demand.

We present these key contributions regarding advanced expert discovery support:

- *Estimation of user reputation based on a context-sensitive algorithm.* Our approach, called *ExpertHITS*, is based on the concept of hubs and authorities in Web-based environments.

- *An approach for community reputation (the hub-expertise of users) influenced by trust relations.* Dynamic link weights are based on trust and user rating influenced by the query context. ExpertHITS is calculated online, thus fully personalized based on the expert-requester's preferences (i.e., the demanded set of skills).

- *Implementation and evaluation of our approach* demonstrating scalability and effectiveness of our proposed algorithm.

9.2 Flexible Involvements of Experts

Once more, we motivate our work with the Expert Web scenario as introduced in Chapter 5. This scenario use case for discovering experts on demand and flexible interaction support is depicted in Figure 9.1. The process model may be composed of single tasks assigned to responsible persons, describing the steps needed to produce a software module. After finishing a common requirement analysis, an engineer evaluates the re-usability of existing work, while a software architect designs the framework. The implementation task is carried out by a software developer, and two software tester evaluate the prototype implementation with respect to functional properties (e.g., coverage of requirements) and non-functional properties (e.g., performance and memory consumption). We assume that the task owners in this process exchange only electronic files and interact by using communication tools.

While various languages and techniques for modeling such processes already exist, for example BPEL, we focus on another aspect in this scenario: *discovery and interactions with trusted experts*. A language such as BPEL demands for the precise definition of flows and input/output data. However, even in carefully planned processes with human participation, for example modeled as BPEL4People activities [1], ad-hoc interactions and adaptation are required due to the complexity of human tasks, people's individual understanding, and unpredictable events. According to Figure 9.1, the software architect receives the requirement analysis document from a preceding step. But if people have not yet worked

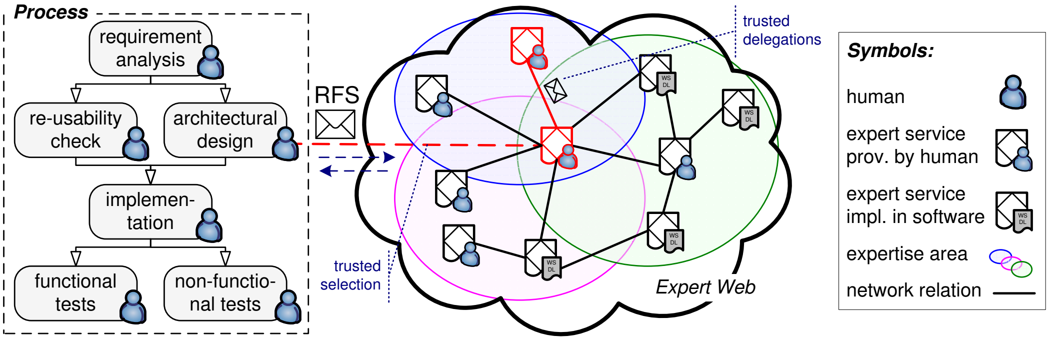


Figure 9.1: Discovering and including experts for online help and suppport.

jointly on similar tasks, it is likely that they need to set up a meeting for discussing relevant information and process artifacts. Personal meetings may be time and cost intensive, especially in cases where people belong to different geographically distributed organizational units. Various Web 2.0 technologies, including forums, Wiki pages and text chats, provide well-proven support for online-work in collaborative environments.

However, several challenges remain unsolved that are addressed in this work. We already highlighted context-aware concepts to involve third-party experts in ongoing collaborations (see Chapter 5 and 8). In this chapter, we deal with an advanced reputation mechanism to support the discovery of experts based on hard skills, *and* social aspects, i.e., trust.

The Expert Web. Here we propose the *Expert Web* consisting of connected experts that provide help and support in a service-oriented manner. The members of this expert web are either *humans*, such as company employees offering help as online support services or can in some cases be provided as software-based services. Applied to enterprise scenarios, such a network of experts, spanning various organizational units, can be consulted for efficient discovery of available support. The expert seekers, for example the software engineers or architect in our use case, send *requests for support*, abbreviated as RFSs. Experts may also delegate RFSs to other experts in the network, for example when they are overloaded or not able to provide satisfying responses. Following this way, not only users of the expert network establish trust in experts, but also trust relations between experts emerge.

On the Emergence of Trust. Traditional rating and ranking models usually neglect social aspects and individual preferences. However, actors in the Expert Web may not be compatible with respect to working style and behavior. As a consequence social aspects need to be considered and require dynamic interaction models. Here, we use *social trust* to support and guide delegations of requests. As discussed in Chapter 4, social trust refers to the flexible interpretation of previous collaboration behavior [7, 37] and the similarity of dynamically adapting interests [38, 134]. Especially in collaborative environments, where users are exposed higher risks than in common social network scenarios [30], and where business is at stake, considering social trust is essential to effectively guide human interactions.

9.3 Expertise Model

In this section we will detail the basic concepts enabling the discovery of experts. Our approach is based on the following idea: given a search query containing the set of relevant skills, who is the expert (i) satisfying these demanded skills and (ii) how well is this expert connected to other people having similar expertise. From the Expert Web point of view, finding the right expert by performing skill matching is not sufficient. We also need to consider whether the expert will be able to delegate RFSs to other peers in the Expert Web.

9.3.1 Personalized Expert Queries

In this work, we define this concept as *expert hubs* that are well-connected (i.e., social network structure *and* connections based on joint collaborations) given a particular query context. Delegation is important in flexible, interaction-based systems because it becomes clear that expert hubs will attract many RFSs over time, thus presenting bottlenecks in terms of processing time needed to work on RFSs. On the other side, being a hub in the Expert Web also means that a person *knows* many other experts in similar fields of interest. We argue that the likelihood of being able to delegate RFSs to other experts greatly increases depending on the *hubness* of a person due to the embedding of a hub in expert areas (e.g., communities or interest groups). The major challenge in this scenario is that hubness needs to be calculated on demand based on a query. The query determines the *context* through the set of skills relevant for discovering experts.

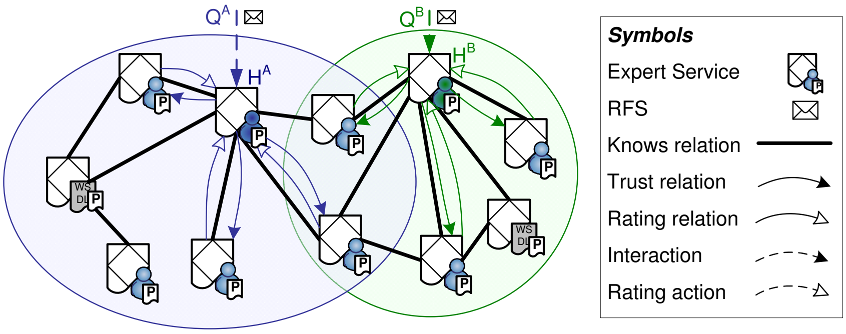


Figure 9.2: Hubs with different areas of expertise.

Let us start formalizing this concept by discussing two scenarios as depicted in Figure 9.2. First, a query (see Q^A or Q^B) is specified either manually by a (human) expert seeker or derived automatically from a given process context, for example a predefined rule denoting that a particular set of skills is needed to solve a problem. The purpose of a query is to return a set of experts who can process RFSs, either by working on the RFSs or delegation. Thus, Q^A would return H^A as the user who is well connected to *authorities* in query context Q^A.

The concept of hubs and authorities has been introduced by Kleinberg [65] to rank Web pages in search queries using the 'HITS algorithm' (Hyperlink-Induced Topic Search). The notion of *authorities* in social or collaborative networks can be interpreted as a measure to estimate the relative standing or *importance* of individuals in an implicitly defined network [65].

There are two influencing factors, i.e., relations, determining hub- and authority scores: (i) how much hubs *trust* authorities (depicted as filled arrows from hubs to authorities) and (ii) *ratings* hubs receive from authorities (open arrows to hubs). Trust mainly influences the potential number of users (e.g., known by H^A) who can process delegated RFSs. On the other hand, receivers can associate ratings to RFSs to express their opinion whether the delegated RFSs fit their expertise. Q^B may demand for a different set of skills. Thus, not only matching of actors is influenced, but also the set of interactions and ratings considered for calculating ExpertHITS (i.e., only the set of RFSs and ratings relevant for Q^B). Note, single *interactions* that lead to trust relations, as well as single *rating actions* that lead to rating relations are not depicted here, but explained in detail in the next section. This approach provides a powerful tool for expert discovery because reputation (for example, within communities) is expressed as hub-expertise by weighting trust relations in personalized scopes (through the query context) and feedback-ratings. Also, we believe that our approach is difficult to cheat on because hub-expertise is influenced by how well hubs are connected to multiple authorities who propagate their expertise back to hubs. We will further elaborate on this concept in the following sections. Before doing so, we discuss the model for expressing skills and areas of expertise.

9.3.2 Skill Model

Our proposed skill model is based on the ACM Computing Classification System[1]. This simple model is sufficient to serve as a basic classification scheme for skills in the computer science domain which is well aligned with the requirements of the previously introduced motivating scenario. More advanced skill or competency models (e.g., ontological systems [10]) are not within the scope of this work. In Figure 9.3, we show an excerpt of a taxonomy that can be used to classify skills in, for example, the software engineering domain.

The basic idea of our approach is to define different weights for each level in the tree (see Figure 9.3(a)). The top-most level (the root node) has the lowest weight since top levels in the skill tree denote broad areas of expertise. The weights increase depending on the tree depth because lower levels contain fine-grained skill and expertise information (specialism). We define the set of levels $L = \{L_0, L_1, \dots, L_n\}$ with $\sum_{i:1 \dots n} w_{L_i} = 1$[2]. All nodes in the skill tree that do not have successor nodes are called *leaf nodes*.

A subset of the tree may be selected by a query Q to discover experts. Thus, the provided query needs to be matched against user profiles to determine how well users match the demanded set of skills. Each node in the tree is called *skill property*. We introduce *query preferences* enabling the expert seeker to express *strong* or *weak* matching preferences of skill properties and *optional* ('nice to have') properties. A strong preference might denote that the expert seeker requires the user to have certain skill properties, whereas weak preferences would express that the expert *should* have as many skill properties. Optional means

[1] ACM Classification System: `http://www.acm.org/about/class/1998/`.

[2] Having all level weights sum up to 1 means that there is a mutual dependency between weights.

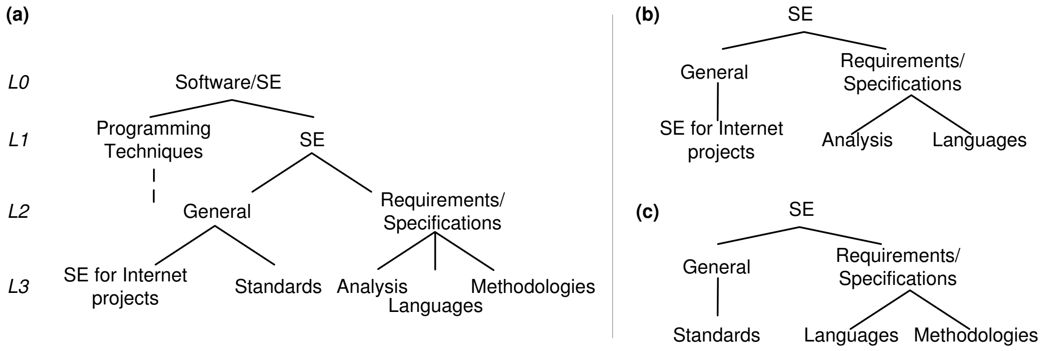


Figure 9.3: (a) Excerpt skill taxonomy in the software engineering domain. (b) illustrates an example query specifying the demanded skills as formulated by an expert seeker. (c) gives an example of a user skill profile.

that higher preferences are given to those experts that have a higher degree of similarity with a set of optional skill properties.

In Figure 9.3(b), an example query is shown that can be formulated as, e.g., `SE for Internet projects` and `[Requirements [Analysis][Language]]` specified within the skill subtree `SE`. For a given user profile, e.g., Figure 9.3(c), matching is performed according to different preferences. Specifying strong preferences for the first part of the query expression would mean no overlap between specified skill property `[SE for Internet projects]` and the user expertise in `Standards`, whereas a weak preference expresses the need to have *some* specialism in a given area (i.e., `[SE [General]]`). Considering the second part of the query expression, strong preferences are calculated as the overlap similarity between the set of skill properties `[Analysis][Language]` and `[Languages][Methodologies]`.

9.4 Expert Discovery

We detail our discovery approach by defining a matching procedure and an algorithm for calculating ExpertHITS. An important aspect of the presented approach is to select interactions for inferring trust relations based on (query) context information. We assume that each interaction (e.g., based on delegated RFSs) is associated with context *tags* based on the skill taxonomy.

Algorithm 9 outlines our approach at a high level, which will be detailed in subsequent sections. First, *matching* is performed based on the query context. In this step, the previously introduced skill model is used to retrieve the set of qualified users. Second, expert hubs are discovered using link and interaction information. As our algorithm is based on Kleinberg's HITS, we calculate hub and authority scores online for each single member of the Expert Web. Third, discovered users are ranked by their *hubness* to determine the best suitable expert.

Next, we define the basic algorithm for matching user profiles and RFSs.

Algorithm 9 Outline discovery approach.

```
*Input:* Given a query context $Q$ to discover expert hubs
*Compute:*
  1. *Find users matching* demanded set of *skills*.
  2. *Calculate hub-expertise* of users given query context $Q$.
     (a) For each user calculate hub score in context $Q$.
     (b) For each user calculate authority score in context $Q$.
  3. *Rank users* by hub score.
*Output:* Ranked experts in $Q$
```

9.4.1 Skill Matching Algorithm

The basic idea is to use a metric to calculate the overlap of two sets A and B as $\frac{|A \cap B|}{m}$ [49]. The presented algorithm supports the notion of strong, weak, and optional matching preferences through alternate approaches for calculating overlap similarities of sets of properties. These preferences have impact on matching of skill properties on lower levels. As mentioned before, all nodes in the skill tree that do not have successor nodes are called *leaf nodes*. For simplicity, we do not consider unbalanced trees or complicated branching structures. Matches at leaf-node level have higher impact on similarities due to the following property: weights increase depending on the tree depth such that $w_{L_0} < w_{L_1} < \ldots < w_{L_n}$. In the following we will derive an algorithm for matching elements which may depict interaction data (tagged RFS-based interactions) and user profiles holding skill information. The function `leafNodes` returns the set of child nodes associated with a given property in the query or the global skill taxonomy tree G_T. The set $|P(L_i)|$ denotes the number of properties in L_i.

Steps 1 - 3 in Algorithm 10 calculate the numerator of the set metric ($|A \cap B|$). The set similarity is divided by the number m based on different matching preferences. As shown in Algorithm 10 (step 4), m is appended to the matching result to obtain similarity scores based on the different preferences $\texttt{pref}_{strong}$, $\texttt{pref}_{weak}$ or $\texttt{pref}_{optional}$ as defined in the following:

$$m = \begin{cases} |\texttt{leafNodes}(q_p(L_i))| \cup |\texttt{leafNodes}(e_p(L_i))| & \text{if } \texttt{pref}_{strong} \\ |\texttt{leafNodes}(G_T(L_i))| & \text{if } \texttt{pref}_{weak} \text{ or } \texttt{pref}_{optional} \\ |P(L_i)| & \text{otherwise} \end{cases} \tag{9.1}$$

9.4.2 Discovery of Expert Hubs

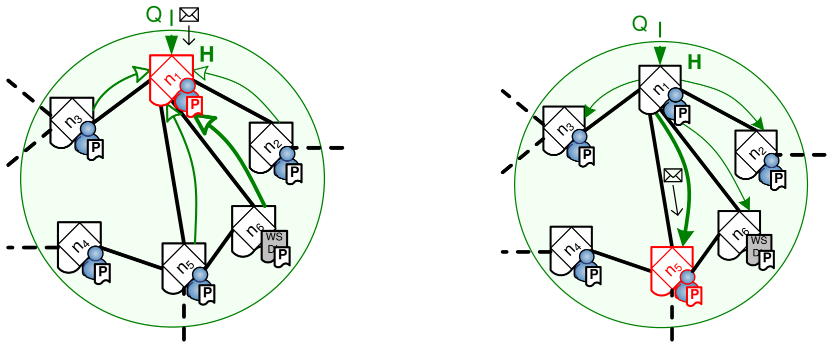
Here we present our expert discovery algorithm that is influenced by trust inference as well as rating mechanisms. Our algorithm accounts for context information and weighted links between actors. Context is utilized by considering relations of experts in different scopes.

Algorithm 10 Topic tree matching algorithm.

```
Input: Given a query context $Q$ containing a set of properties $q_p$ and elements $E$
Compute:
  1. Get all elements $e \in E' \subseteq E$ whose properties provide a minimum match of topics.
  2. Extract topic tree matching query root node.
  3. Iterate through each level and calculate overlap similarity of property in query at current level $i$. Given current property $q_p(L_i)$:
     (a) If $\texttt{leafNodes}(q_p(L_i)) \neq \emptyset$ then do $|\texttt{leafNodes}(q_p(L_i))| \cap |\texttt{leafNodes}(e_p(L_i))|$.
     (b) If $\texttt{leafNodes}(q_p(L_i)) = \emptyset$ and weak preference then use $|\texttt{leafNodes}(e_p(L_i))|$.
     (c) Otherwise perform $q_p(L_i) \cap e_p(L_i)$.
  4. Divide similarity by $m$ and append score with $w_{L_i}$ to previous score sum.

Output: Ranked elements according to similarity
```

Thus, the goal of our algorithm is to find hubs with respect to context. In the following, we discuss the basic flow of *actions* in the Expert Web. The actions include delegations of RFSs, ratings of requests, and advanced delegation patterns (see Chapter 5). First, we discuss the discovery and selection of expert hubs and authorities (Figure 9.4) followed by the definition of delegation patterns and rating (Figure 9.5).



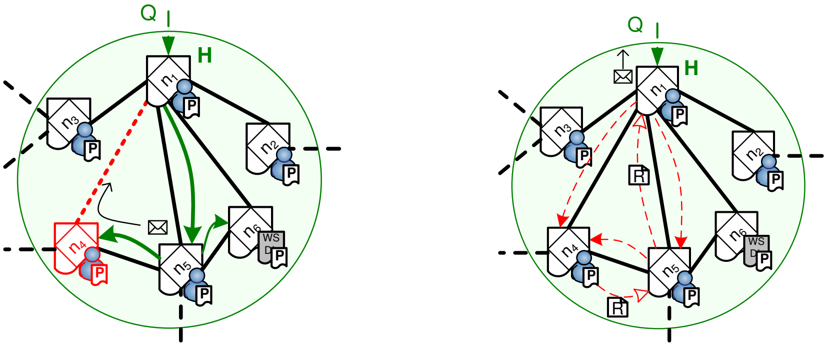
(a) Discovery of expert hub.

(b) Trusted selection of authority.

Figure 9.4: Interaction and discovery model.

Hub discovery. Let us assume that a query Q is specified to discover an expert hub (see Figure 9.4(a)). Every query influences the set of prior ratings (arrows pointing to n_1) and interactions (i.e., actions) that need to be considered when calculating hub- and authority scores. Consider the query context Q comprising actions related to the demanded set of skills. Note, the previously defined matching algorithm is used to select related actions. In this case, n_1 has been discovered as the entry point denoted as H

Delegation actions. In Figure 9.4(a), user n_1 receives an RFS issued towards the Expert Web. Since n_1 represents the hub expert, n_1 may decide to delegate the request to one of its neighbors n_2, n_3, n_5, n_6, which can be discovered through `knows` relations[3] (Figure 9.4(b)). In our Expert Web application scenario, `knows` is a bidirectional relation between users. A relation becomes active if both users acknowledge that they are connected to each other (n_2 `knows` n_1 *and* n_1 `knows` n_2), a simple yet effective mechanism to support growth in social networks (e.g., newcomers and bootstrapping problem) while preserving user control. Note, `knows` relations do not contain context related information such as tags. The context of interactions is derived from delegated RFSs (tags or type of RFS classified by using the skill taxonomy). To support growth in networks (e.g., how can newcomers become members of communities), we introduce an advanced interactions pattern in the Expert Web depicted by Figure 9.5(a).



(a) Delegation of RFS.

(b) RFS reply and rating.

Figure 9.5: Advanced interaction patterns and rating.

Triad delegation pattern. An authority may need to delegate an RFS received from the hub to somebody who is known to the authority, but not the hub. This pattern is shown in Figure 9.5(a). Hub n_1 delegates an RFS to n_5, which is in turn delegated to n_4 and, thus, being responsible for processing the RFS.

If ties (i.e., through `knows` relations) between the pairs (n_1, n_5) and (n_5, n_4) exist, it is likely that n_4 will attempt to establish a connection to n_1 as well. This concept is known as *triadic closure* in social networks [128] and can be applied to support interaction patterns in service-oriented systems. The *triad interaction pattern* (see [116]) enables n_4 to connect to hubs and helps increasing its authority in the Expert Web. As mentioned previously, `knows` is a bidirectional connection and needs to be acknowledged by n_1.

Rating procedure. An RFS is delivered back to the expert seeker from the Expert Web; i.e., the selected hub n_1 depicted in Figure 9.5. The main argument of our model is to determine those hubs that are well embedded in expertise areas (e.g., communities). Thus, the hub-score should be influenced by feedback ratings denoting the level of satisfaction of authorities. Ratings are subjective opinions of authorities with respect to RFSs received from hubs, i.e., whether RFSs fit the expertise area of authorities. In the final step, RFSs are rated (see dashed open arrows) expressing the precision of received delegations. Indeed,

[3] The knows property in FOAF profiles can be used for discovering social relations; see `http://xmlns.com/foaf/spec/` issued at 2 Nov. 2007.

such ratings are also given to RFSs traversing the Expert Web through triad delegation patterns. Given the scenario in Figure 9.5, automatic propagation of ratings (e.g., if a delegated RFS from n_1 to n_5 was further delegated to n_4) is currently not considered in our model. Thus, n_4 rates the RFS received from n_5 and similarly n_5 from n_1.

Trust updates. Trust relations, based on experts' behavior in terms of reliability, availability, or RFS processing successes, are periodically updated with recent interaction data. Those interactions (reflected by filled dashed arrows) are aggregated to interaction metrics that are interpreted by pre-defined rules to infer trust. The detailed mechanisms has been studied in Chapter 4.

Formalized Model

In the following, we discuss the formal model for our proposed expertise ranking algorithm consisting of two components (i) hub score $H(n_i; Q)$ of user n_i in query context Q and (ii) authority score $A(n_j; Q)$ of user n_j in the same query context Q.

- $H(n_i; Q)$: Hub score of user n_i acting as a reliable entry point to the Expert Web brokering RFSs to authoritative users. Hubs are identified based on the demanded expertise, `knows` relations connecting n_i to other experts *and* feedback ratings received from prior delegations.

- $A(n_j; Q)$: Authority score of user n_j. Authorities are skilled users (experts) that are connected to influential hubs. In our model, authority means that users process RFSs received from hubs in a reliable, trustworthy manner.

$$H(n_i; Q) = \sum_{n_j \in \texttt{knows}(n_i)} w^Q_{n_j n_i} A(n_j; Q) \tag{9.2}$$

$$A(n_j; Q) = \sum_{n_i \in \texttt{knows}(n_j)} w^Q_{n_i n_j} H(n_i; Q) \tag{9.3}$$

An important factor in our model is the estimation of link weights. Table 9.1 gives a description based on which parameters the weights are calculated.

Symbol	Description
$\texttt{knows}(n_i)$	The set of users known by n_i.
$w^Q_{n_i n_j}$	Trust influences the delegation behavior of hubs by selecting authorities based the success of interactions; in our example successfully delegated and processed RFSs.
$w^Q_{n_j n_i}$	Denotes the connection strength of an authority n_j to hub n_i. In other words, $w^Q_{n_j n_i}$ influences the 'hubness' of n_i. The weight can be calculated using information from ratings given by n_j to RFSs received from n_i.

Table 9.1: Hubs and authorities in the Expert Web.

The weight $w^Q_{n_i n_j}$ can be interpreted as how much n_i trusts n_j in processing RFSs in a reliable manner. The weight can be estimated as

$$w^Q_{n_i n_j} = \frac{\text{Successful delegations from } n_i \text{ to } n_j}{\sum\limits_{n_x \in \mathtt{knows}(n_i)} \text{Successful delegations from } n_i \text{ to } n_x} \tag{9.4}$$

9.5 Evaluation

In our experiments we focus on the performance of ExpertHITS as well as the influence of trust and ratings on hub/authority scores. In this work, we do not deal with performance issues due to network delay or end-to-end characteristics of the entire system. Here we focus on ExperHITS calculation time under different conditions.

Experimental Setup. In all our tests we used a machine with Intel Core2 Duo CPU 2.50 GHz, 4GB RAM, running Java 1.6 and an OSGi Java container for hosting services. A query service has been implemented on top of the HPS Framework [106]. We use trust mining and metric calculation capabilities available as services by the VieTE (Vienna Trust Emergence) framework [117] to construct graphs based on user relations and trust. The ExpertHITS algorithm has been implemented on top of a Java-based graph modeling toolkit[4].

Data Generation. The approach we take is to generate artificial interaction data imitating real collaboration environments. For this purpose, we adopt the *preferential attachment* method [97] which provides a realistic model for science collaboration scenarios. Specifically, a collaboration network is modeled as an undirected graph $G = (N, E)$ comprising a set of nodes N and edges E establishing connections between nodes. The probability of establishing a new connection to a given node is proportional to its degree distribution. Using this basic network structure, we generate interactions (delegations and ratings) associated with edges. Assuming a scale free network with power law distribution, hubs play a central role, thereby generating a large amount of delegations. This behavior is taken into account when generating artificial interactions by estimating that 80% of delegations are initiated by about 20% of network users (i.e., immitating hubs).

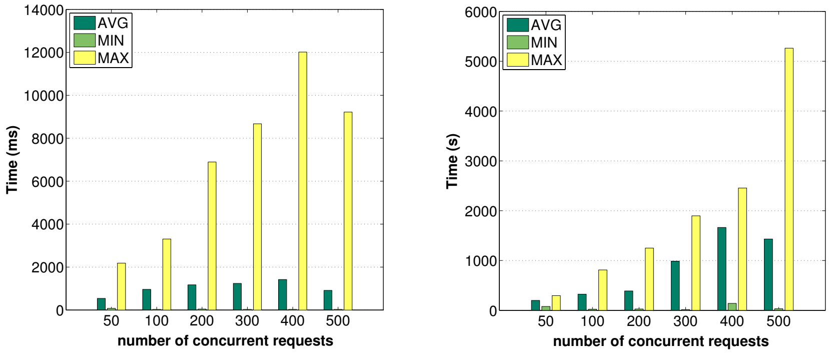
Results. Complexity is a crucial factor in order to support personalization of queries. The complexity for computing ExpertHITS is $\mathcal{O}(|N| * it)$, $|N|$ representing the number of nodes in the graph and it the number of iterations until the algorithm converges. We analyze different networks comprising actors and interactions that have already been matched with a specific query context (see Table 9.2).

network characteristics	ExpertHITS computation time
Small-scale: 100 nodes, 400 edges	$\approx$ 60 ms
Medium-scale: 1000 nodes, 4000 edges	$\approx$ 600 ms
Large-scale: 10000 nodes, 40000 edges	$\approx$ 12100 ms

Table 9.2: ExpertHITS computation performance.

[4] JUNG: http://jung.sourceforge.net/

ExperHITS can be computed in a sufficient amount of time scaling up to large networks (i.e., 10000 nodes). Notice, these networks represent already filtered subgraphs based on the query context. We believe that this assumption is realistic considering the targeted Expert Web consisting of professionals. The system must be able to handle multiple requests simultaneously. We analyze the performance of ExpertHITS under different load conditions. At this stage, we focus on small-scale (100 nodes) and medium-scale (1000 nodes) networks. Figure 9.6(a) and Figure 9.6(b) show the results given 50-500 concurrent requests to calculate ExpertHITS. A queue holds instances of the constructed network. A thread pool instantiates worker threads to calculate personalized ranking scores based on query preferences. Small-scale networks can be processed in a real-time manner requiring in our experiments in the worst case (MAX values) up to 12 seconds. On average, 17 seconds can be expected under different load conditions (50-500 concurrent requests). The results of medium-scale networks are shown in Figure 9.6(b) and compared with small-scale networks in Figure 9.7. Computing ExpertHITS in such networks takes up to several minutes when serving concurrent requests (i.e., on average 390s at a load of 200 requests). Load conditions in the range between 300-500 concurrent executions of the algorithms results on average in response times between 15-25 minutes. Given our initial online help



(a) Small-scale.

(b) Medium-scale.

Figure 9.6: Concurrent request processing time of ExpertHITS.

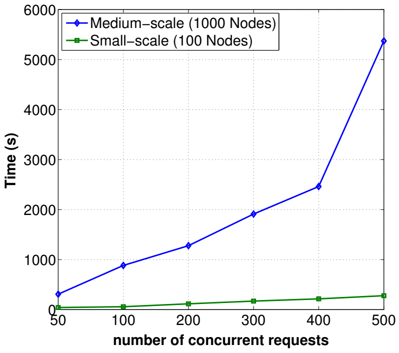
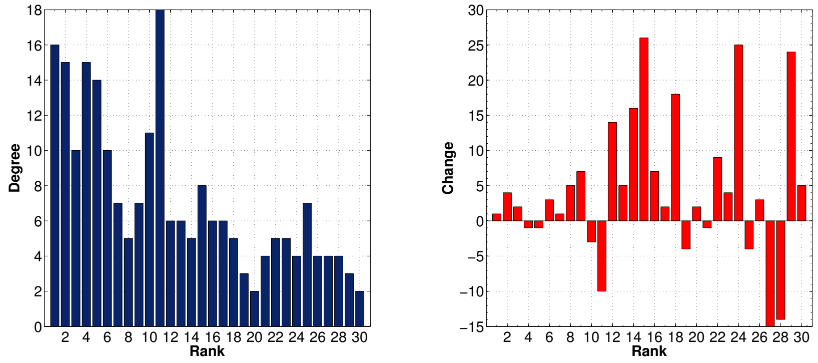


Figure 9.7: Comparison of total processing times.

and support example, we believe it is sufficient to compute ExpertHITS in this magnitude because illustrated processes in software engineering do not demand for hard computational (time) constraints. Scalability and reduced processing time can be achieved by using multiple servers and load balancing mechanisms. These mechanisms are subject to our future work and performance evaluation.



(a) Node degree.

(b) Ranking change.

Figure 9.8: Impact of ExpertHITS on rankings.

To test the effectiveness of ExpertHITS, we performed experiments to study the impact of ratings and trust on expert rankings. In Figure 9.8, we show the top-30 ranked experts in a small-scale network (100 nodes). Results are sorted based on the *position* within the result set (horizontal axis). Figure 9.8(a) shows the degree of network nodes and Figure 9.8(b) ranking changes obtained by comparing ranking results using the HITS algorithm without taking trust or ratings into account. Specifically, $pos(n_i)_{HITS} - pos(n_i)_{ExpertHITS}$ returns the absolute ranking change of n_i in a given result set.

In Figure 9.9, we show the average rating of each ranked node; average rating of node n_i received from its neighboring nodes divided by the expected rating. We define *quality* as the aggregated trust weights. Quality is calculated as $\sum_{n_j \in \texttt{knows}(n_i)} \sum_{n_k \in \texttt{inlink}(n_j)} w_{n_k n_j}$.

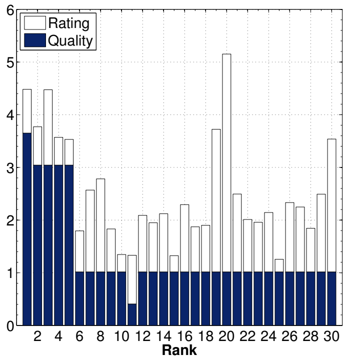


Figure 9.9: Ratings and quality.

Quality measures the overall trust in node n_j. In Figure 9.9, we see that all nodes within the top segment received high ratings given a high degree of links which is the desired property of ExpertHITS. Some nodes are demoted (negative ranking change) since the node (e.g., see 11) has received low ratings even though the node has a high degree of links. On the other hand, nodes get promoted (positive ranking change) if they exhibit sufficient high ratings (see 15) or high quality (see 20 which was promoted a few positions only due to limited degree). Overall, ExpertHITS exhibits the demanded properties of promoting well-connected and rated hubs, thereby guaranteeing the discovery of reliable entry points to the Expert Web.

9.6 Discussion

In this chapter, we introduced a new approach for supporting trust- and reputation-based expert discovery. Unlike traditional models found in process-centric environments, we proposed the combination of preplanned process steps *and* ad-hoc activities to solve emergent problems in distributed collaboration environments. Our approach is based on the HPS concept enabling knowledge workers to offer their skills and expertise in service-oriented systems. Expert discovery is greatly influenced by (behavioral) trust and reputation mechanisms. We demonstrated a novel approach for estimating expert reputation based on link structure and trust relations. Trust information is periodically updated to capture dynamically changing interaction preferences and trust relations. We have shown that ExpertHITS can be computed in an online manner, thereby enabling full personalization at runtime. Existing approaches in personalized expertise mining algorithm typically perform offline interaction analysis. Our empirical evaluations have shown that ExpertHITS exhibits the desired properties; trust and rating weights influence hub- and authority scores. These properties ensure that our algorithm discovers experts which are well-connected to other experts.

CHAPTER 10

Trust-based Adaptations in Complex Service Systems

Outline. We discuss the meaning of trust for dynamic adaptations in service-centric systems. For that purpose, we highlight mechanisms for flexible information disclosure in collaboration scenarios.

Contents

10.1 Motivation

The way people interact in collaborative and social environments on the Web has evolved in a rapid pace over the last few years. Services have become a key-enabling technology to support collaboration and interactions. Pervasiveness, context-awareness, and adaptiveness are some of the concepts that emerged recently in service-oriented systems. A system is not designed, deployed, and executed; but rather evolves and adapts over time. This paradigm shift from closed systems to open, loosely coupled Web services-based systems requires new approaches to support interactions.

Trust emerging in specific scopes, relies on previous interactions, and evolves over time. Accounting for dynamically changing trust relations when selecting people for communication or collaboration, services to be utilized, and resources to be applied, leads to more efficient cooperation and compositions of human- and software services [111]. Some typical aspects that require run-time adaptation in mixed service-oriented systems include:

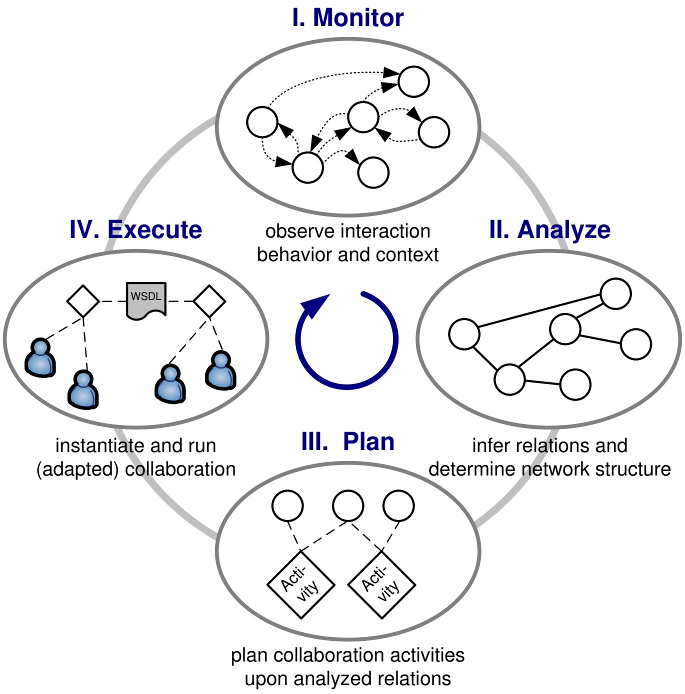

I. Monitor
observe interaction behavior and context
II. Analyze
infer relations and determine network structure
III. Plan
plan collaboration activities upon analyzed relations
IV. Execute
instantiate and run (adapted) collaboration
WSDL
Acti-vity
Acti-vity

Figure 10.1: Adaptation approach for mixed service-oriented systems.

- *Discovery of Network Members and Resources.* In many networks, for example social networks, the discovery and selection process relies on matching of user profiles and resource features that are mainly static. In contrast, utilizing periodically updated trust relations better accounts for varying user preferences and avoids lookup based on stale information.

- *Access to and Sharing of Information.* Traditional approaches to access rights management are based on manually assigned static user roles. However, the user is often not able to keep track of configurations in complex networks such as dynamically changing roles.

- *Coordination and Compositions.* Especially in flexible environments, compositions of humans and services cannot only rely on static structures, but have to be flexibly adapted based on their run-time behavior.

- *Interaction Policies and Patterns.* In common enterprise networks, policies and interaction patterns describe and restrict communication paths between network members. Therefore, periodic adaptation upon ongoing collaborations enable optimizations according to the outcome of interactions.

We introduce a general approach to deal with these concerns, enabling trust-based adaptation of complex network structures (Figure 10.1). This concept follows an adopted version of the 'MAPE' cycle as introduced in Chapter 3. MAPE, incorporating fundamental concepts from control engineering, describes a cycle consisting of four phases, which are ***Monitor***, ***Analyze***, ***Plan*** and ***Execute***. Periodically running through these four phases establishes a kind of environmental feedback control, and, therefore, allows to adapt to varying circumstances. This aspect enables us to address aforementioned challenges. Briefly explained again, in the *Monitoring Phase* our system observes interactions, in particular communication, coordination, and execution events among network members in their respective situations. In the *Analyzing Phase* logged interactions are used to infer relations and to determine network structures. For this purpose, domain-dependent interaction metrics and quality criteria are calculated and interpreted. The following *Planning Phase* covers the preparation of new collaborations, for instance, discovery, ranking and selection of network members, services, and resources. In the *Execution Phase* either new collaborations are instantiated, or existing scenarios adapted according to feedback from prior observations. In that phase network members interact to perform planned activities. This closes the cycle.

Previously, we introduced methods and algorithms that are applied in the monitoring and analyzing phases of the MAPE approach for inferring trust by interpreting and weighting interactions [111]. In this chapter, we describe the realization and major design decisions of frameworks that support adaptations in complex service-oriented networks. We present the following contributions:

- *Trust-based Adaptation in Complex Systems.* We focus on complex networks of human and service actors. In that environment, we describe the emergence of trust upon interactions, and discuss a self-adaptive approach as well as typical concerns.

- *Realization and Implementation Aspects.* We discuss the support and realization of one representative mixed complex network example. In that use case, information sharing among network members is adapted by accounting for dynamically emerging trust relations.

- *Evaluation and Discussion.* We evaluate our Web services-based implementation with performance studies under realistic conditions.

10.2 The Science Collaboration Scenario

A typical environment for applying trusted information sharing is a *science collaboration network*. It comprises scientists, members from national and international research labs, and experts from the industry. Collaboration is supported by modern service-oriented architectures that realize centralized people registries and profile management, communication services, and data sharing facilities. Network members collaborate to address challenging research questions and to reach higher impact of scientific disseminations. They conduct joint project proposals, perform distributed software prototyping, and data analysis and visualization. Furthermore, certain participants can provide their support in a service-oriented manner. For instance, they offer document review services, or data analysis services, and interact through precisely predefined interfaces. We utilize the previously introduced Human-Provided Services (HPS) framework [106] to embed humans acting as services using SOA concepts. This includes WSDL descriptions of interfaces, central registries, SOAP-based interactions, and sophisticated logging facilities.

10.2.1 Emerging Trust Networks

We demonstrated the (semi-)automatic flexible determination of trust [111] in the above-mentioned service-oriented collaboration environment in previous chapters. Briefly, our approach relies on the observation of fundamental interactions, such as SOAP-based communication, coordination or execution messages. People interact and use services when conducting activities. Figure 10.2 depicts this fundamental concept. Network members collaboratively perform activities of different types. These activities structure relevant contextual information, including involved actors, goals, temporal constraints, and assigned resources. So, we conclude that an activity holistically captures the context of interactions between participants [111]. Several activity contexts are aggregated to uniform scopes, e.g., all activities of a specific type (activity scope), or all activities belonging to a certain project (project scope). Trust emerges from interactions and manual ratings of collaboration partners within those scopes. For instance, trust can rely on the responsiveness and reliability of collaboration partners, as well as on their collected experiences and skills. As shown in Figure 10.2, trust is represented by a directed relation from one network member n_i (the *trustor*) to another one n_j (the *trustee*), and relies on prior cooperative behavior in a given scope. These trust relations are determined by periodically analyzing and interpreting observed interactions and ratings of partners. For example, the collaboration of network members n_1, n_2, n_3, and n_4 in different scientific dissemination activities a_1 and a_2, leads to the establishment of trust in one uniform 'dissemination scope'. Finally, a scale-free complex network emerges from cooperations in typical research collaborations as investigated by [97].

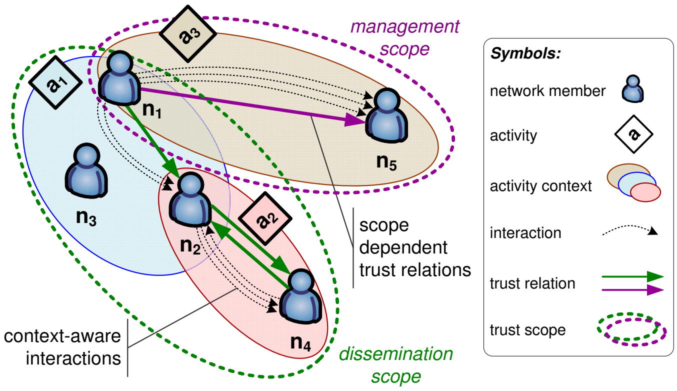


Figure 10.2: On the emergence of trust.

10.2.2 On Trusted Information Sharing

In a science collaboration network scenario, understandably no member will share novel, yet unpublished, ideas carelessly. However, information sharing is essential to discover new collaboration opportunities. The challenge is to enable sensitive information sharing, that adapts and restricts the view on information with respect to changing trust relations. Therefore, we introduce the concept of *trusted information sharing*. This concept provides the means to share information, e.g., paper drafts, recently submitted papers, or project documentation, only with trusted network members who have demonstrated their reliable and dependable behavior before. In this case, trust reflects a probability measure of future collaboration successes, and therefore, potential benefits from collaborations.

As depicted in Figure 10.3, trusted information sharing is bound to trust scopes. For instance, if member n_1 established trust in n_5 in the management scope (because they jointly performed several project management activities successfully), n_5 is allowed to access n_1's data about referees' contact details, planned future projects, and personal organizational details. However, no information dedicated to other scopes, such as scientific dissemination, is shared. Hence, information sharing is restricted to mandatory information in particular scopes.

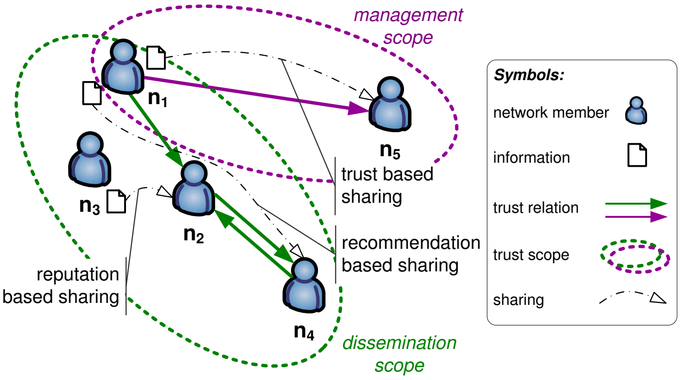


Figure 10.3: Trust concepts utilized for trusted information sharing.

As trust relations emerge dynamically based on interaction behavior of people, the amount of shared information is periodically adapted by the system and, in the optimal case, needs no further manual intervention of users. However, this approach works best in environments with flat (or practically no) hierarchies, where people may decide completely on their own about conditions for information sharing. In enterprise collaborations, with pre-determined communication paths and static role models, mechanisms that override trust-based sharing are required. But here, we focus on the depicted science collaboration network that consists of people with equal roles, rights and aims. We identified three fundamental trust concepts to enable trusted information sharing in the described environment:

Sharing based on Personal Trust Relations. Activity relevant artifacts are shared in a scope to different extent (views), according to the degree of trust between network members. For instance, in Figure 10.3 n_1 grants access to n_5 to information in the management scope.

Sharing based on Recommendations. In case of sparse trust networks, or low connectivity of certain members, sharing based on personal relations only is limited. Second-hand opinions, called recommendations, are utilized to overcome this problem. For instance, n_1 trusts n_2, and n_2 trusts n_4 because of successful previous collaborations in the dissemination scope. If these successes rely on the compatibility of each member's working style, there is a high probability that n_1 might establish trust to n_4 upon future interactions (for transitive trust propagation see [46]). Hence, to facilitate the establishment of trust relations, n_1 is encouraged to share pieces of information with the unknown member n_4. Sharing of data, such as parts from the personal profile, introduces n_1 to n_4 and supports the bootstrapping of future collaborations [134].

Sharing based on Reputation. If network members are trusted by several partners in the same scope, (i.e., they have more than one trustor), reputation can be determined. For instance, n_2 is trusted by n_1 and n_4. Therefore, network member n_3, who has not established trust in others yet, can rely on this reputation (inferred from single trust relations). So, n_3 can either allow n_2 to access parts of his personally managed information (passive sharing), or by pushing information towards n_2 (active sharing).

10.2.3 Context Model and Shared Information

We adapt our previously shown activity model as depicted in Figure 10.4 to capture the context of interactions, and to distinguish and categorize interaction behavior with respect to different situations. It reflects the relationships between managed information in social and collaborative networks, including the introduced science collaboration network scenario. Briefly, this model comprises the aggregated information from various supporting services, such as community member profiles, calculated interaction and trust metrics, personal information, and involved activities.

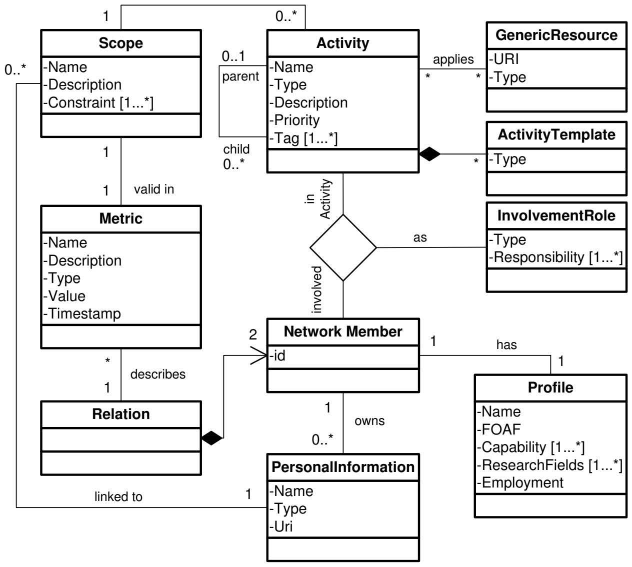


Figure 10.4: Interaction context model and shared information.

We enable the sharing of all of this information. Hence, in contrast to traditional approaches, such as P2P file sharing that focuses on sharing of document-based information only, we also allow sharing of social information. Besides personal data, this includes profiles, member relationships, activity involvements, regular co-workers, or collaboration performance determined by previous interactions.

10.3 Design and Architecture

The most fundamental use case of trusted information sharing is as follows: A network member n_i (the trustor) has established trust in his collaboration partner n_j (the trustee) due to previous cooperative behavior in a specific scope. Therefore, the owner (trustor n_i) of some information that is identified by an uri is willed to share this information with his trustee n_j.

We distinguish between two *modes of sharing*: (i) Activity-centric sharing accounts for the currently jointly processed activity of n_i and n_j. Therefore, information is shared to foster ongoing collaborations. (ii) Scope-centric sharing is about information sharing due to trust in a scope, but without accounting for a concrete activity. This kind of sharing is useful to facilitate future collaborations, i.e., the creation of new joint activities.

Besides the modes we distinguish two different *sharing styles*: (i) Active Sharing pushes information to actual or potential collaboration partners (depending on the sharing mode), e.g., a call for paper via announcement services. (ii) Passive Sharing grants access to personal information when requested by other network members, e.g., when the collaboration network is searched for dissemination opportunities. We focus on the latter kind of sharing style that can be understood as a dynamic access control system.

10.3.1 Sharing Framework

This section details the structural view of the framework (components) and the dynamic aspects (invocations) describing the mode of operation.

10.3.1.1 Structural View

The major components of our framework and their connections are briefly shown in Figure 10.5. The backend services comprise one or more *Information Repositories* that hold various kinds of information, encoded in XML and defined by XML schemes (XSDs). An *Information Catalog* enables users to link information from repositories to sharing scopes. Activities, as introduced in our motivating scenario, are managed by an *Activity Management Service* and act as the glue for multi-dimensional collaboration data (see the context model in Figure 10.4). Especially trust relations that emerge from interactions between users during the execution of activities, are provided by the *Trust Network Provider*. A *Sharing Rule Management Service* provides trust requirements for information sharing, e.g., a minimum degree of personal trust or reputation, and the *Sharing View Management* stores transformation scripts (XSLTs) to customize the view on XML-encoded information. The *Sharing Proxy* utilizes all the aforementioned SOAP-based services and restricts information based on sharing views picked by evaluating sharing rules. Technically, this is realized by transforming XML data through applying XSLTs depending on trust relations

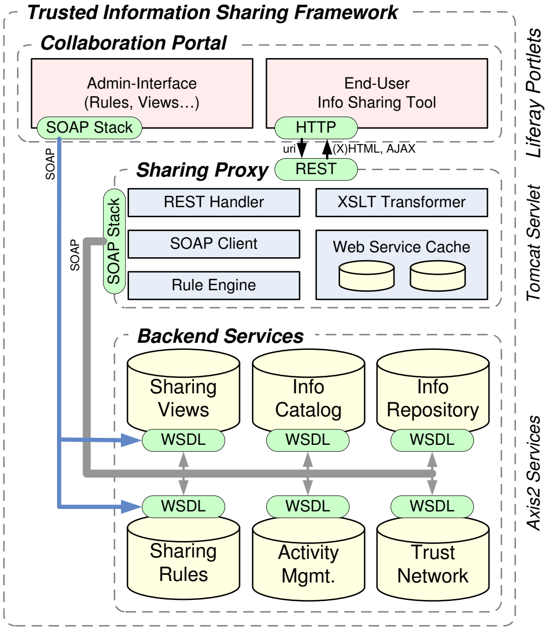


Figure 10.5: Architectural overview of the sharing framework.

between information owner and requester. Higher trust allows more details to be shared. In the end-user collaboration portal, an *Information Sharing Tool* is provided that communicates with the Sharing Proxy via a REST-style interface, and allows to create, read, update and delete (CRUD) shared information. This includes adding new information to repositories (e.g., document stores) and registering this information in the Information Catalog. An *Administrator Interface* enables the configuration of sharing rules and views (XSLTs), as well as the registration of new information types (XSDs).

10.3.1.2 Fundamental Mode of Operation

We describe the interplay of the components to cover the fundamental use case of trustworthy sharing of a *particular* information (i.e., that is already referenced by an uri), of the owner n_i with the requester n_j. Let us assume, n_i explicitly publishes the uri of an XML file in a public area of the collaboration platform. User n_j wants to retrieve this information through the REST interface of the Sharing Proxy, and view in his Browser. That is the point, where *trustworthiness* comes into play. The sequential interactions of the single components are depicted in Figure 10.6. The process starts with retrieving registered meta-data for the given information, including the owner and valid scopes of sharing. After that, joint scopes are requested from the Activity Management Service, i.e., the scopes of all joint activities. Then, the sharing rules of the information owner are retrieved, as well as existing trust relations in the potential sharing scopes. The Sharing Proxy picks the sharing rule that results in the least restrictive information. This means sharing relies on the tightest available trust relation between owner and requester. According to the picked rule, the corresponding XSLT script from the Sharing View Management Service is requested,

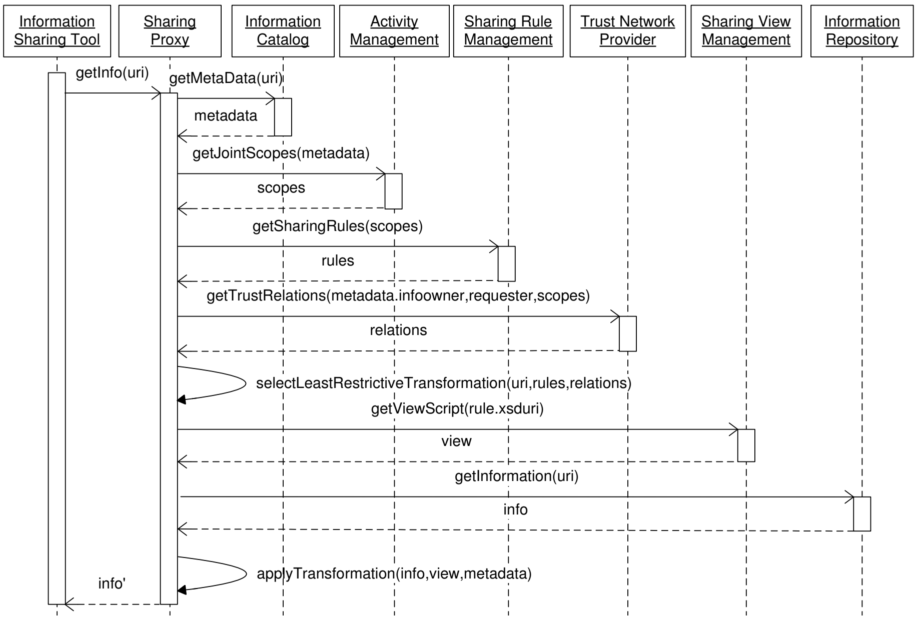


Figure 10.6: Mode of operation of the sharing framework.

as well as the initially requested information from the Information Repository. Finally, the requested information is dynamically transformed to its trustworthy view and delivered to the requester.

10.3.2 Implementation Details

The information sharing framework, depicted in Figure 10.5, is designed as distributed service-oriented system, where the single components are implemented as Web services with SOAP and REST interfaces. In this section we highlight implementation decisions, that are further discussed in the evaluation part of this chapter.

Sharing Proxy Interface. In contrast to the other components, the Sharing Proxy is not implemented as a SOAP-based Web service, but as a Servlet with a REST-style interface [34]. On the one side, this fact simplifies the integration with the collaboration portal (JSR-168 portlets[1]), on the other side, processing pure HTTP requests deem to be more scalable than SOAP messages. Resource repositories are typical applications for RESTful interfaces, where each resource is explicitly identified by a corresponding uri. The requester is identified by HTTP authorization in each request, therefore no further parameters than the uri of the information of interest is required to enable trusted information sharing. Table 10.1 summarizes the available RESTful operations of the Sharing Proxy. The uri for each resource is composed of the uri of the proxy servlet with additional scopeId, activityId, memberId, and optional infoURI. If the requester omits the infoURI, a collection of all information (with optional type selection) identified by the given uri is returned (*/listInfos&type=XSD*). Restrictions on scopes, activities, and members are not

[1] http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html

Operation	servletURI/scopeId/activityId/ memberId/listInfos&type=xsd	servletURI/scopeId/activityId/ memberId/infoURI
GET	get all information uris (collection overview)	get specific info identified by uri
PUT	–	replace/update existing information (only with same XSD))
POST	create new information (following existing XSD)	–
DELETE	–	delete specific info (if the requester is the registered owner)

Table 10.1: Sharing Proxy REST-style interface.

mandatory, and can be replaced with *anyScope/anyActivity/anyMember*. For instance, links to all shared paper drafts of any community member in the scope of ‘scientific dissemination’, can be found in *servletURI/disseminationScopeId/anyActivity/anyMember/listInfos&type=paperdraft.xsd*.

Trust Network Provider Interface. Network members retrieve data about connected neighbors in a system-managed trust graph, and can search for users by name and profile data (similar to a lightweight service registry). Furthermore, the service offers information about someone’s trust relations, second-hand recommendations, and third-party reputation. Design details of the Trust Network Provider can be found in the appendix of this work. For more information on sharing social relations see Chapter 11.

Information Definitions and Repository. Shared information has one of the following origins: (i) information that is manually managed by users, such as documents, notes, and source code in external repositories; and (ii) information that is generated and managed by the system according to the context model. All information structures are pre-defined by XSDs, provided by administrators of the platform. Listing 10.1 shows exemplarily a paper draft XSD that is suitable for the academic research network scenario (and further used in the evaluation part).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:p="http://www.infosys.tuwien.ac.at/tis/papercon"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:import namespace="http://www.infosys.tuwien.ac.at/tis/papercon"
      schemaLocation="paperconcepts.xsd"/>
   <xsd:element name="paperdraft" type="tpaperdraft"/>
   <xsd:complexType name="tpaperdraft">
      <xsd:sequence>
         <xsd:element name="title" type="xsd:string"/>
         <xsd:element name="author" type="p:author" maxOccurs="unbounded"/>
         <xsd:element name="contact" type="xsd:string"/>
         <xsd:element name="category" type="p:category" maxOccurs="unbounded"/>
         <xsd:element name="keywords" type="xsd:string" maxOccurs="unbounded"/>
         <xsd:element name="abstract" type="xsd:string" minOccurs="0"/>
         <xsd:element name="body" type="xsd:string" minOccurs="0"/>
         <xsd:element name="lastChangeAt" type="xsd:dateTime"/>
         <xsd:element name="linkedRes" type="xsd:anyURI" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
   </xsd:complexType>
</xsd:schema>
```

Listing 10.1: XSD for information about paper drafts.

Information Registration. Users register each item of information that they intend to share in the Information Catalog (however, this can be automatized with more advanced tool support). By creating catalog entries, they link information (identified by uris of XML data and corresponding XSD(s)) to scopes. In this way, users decide on their own which information can be shared in which scopes. Listing 10.2 shows an excerpt of the schema of such catalog entries.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
   <xsd:element name="entry" type="tEntry"/>
   <xsd:complexType name="tEntry">
      <xsd:sequence>
         <xsd:element name="registeredName" type="xsd:string"/>
         <xsd:element name="infoXSD" type="xsd:anyURI"/>
         <xsd:element name="infoURI" type="xsd:anyURI"/>
         <xsd:element name="owner" type="xsd:anyURI"/>
         <xsd:element name="scope" type="xsd:anyURI" maxOccurs="unbounded"/>
         <xsd:element name="mode" type="tmode"/>
         <xsd:element name="registeredAt" type="xsd:dateTime"/>
         <xsd:element name="updatedAt" type="xsd:dateTime"/>
         <xsd:element name="comment" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
   </xsd:complexType>
   <!-- ... -->
</xsd:schema>
```

Listing 10.2: Catalog entry schema excerpt.

The main advantage of separating the actual information (e.g., paper drafts) from sharing management data (e.g., scope of sharing, owner, mode) is that the same information can be linked to different scopes, and links can be dynamically modified without affecting the actual information (*separation of concerns*). The schema (Listing 10.2) is designed to enable multiple types of search queries, such as retrieving shared information in a scope, of a specific type (XSD), of a particular user, or combinations of these parameters.

Sharing Rule Definitions. In addition to catalog entries, users who want to share information also define sharing rules that account for dynamically changing trust relations.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"...>
   <xsd:element name="rule" type="tRule"/>
   <xsd:complexType name="tRule">
      <xsd:sequence>
         <xsd:element name="owner" type="xsd:anyURI"/>
         <xsd:element name="validScope" type="xsd:anyURI" maxOccurs="unbounded"/>
         <xsd:element name="applyOnType" type="xsd:anyURI"/>
         <xsd:element name="condition" type="tCondition"/>
         <xsd:element name="applyXSLT" type="xsd:anyURI"/>
         <!-- ... -->
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="tCondition">
      <xsd:sequence>
         <xsd:element name="trust" type="tnValop" minOccurs="0"/>
         <xsd:element name="recommendation" type="tnValop" minOccurs="0"/>
         <xsd:element name="reputation" type="tnValop" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <!-- ... -->
</xsd:schema>
```

Listing 10.3: Sharing rule schema excerpt.

According to the excerpt in Listing 10.3, users define in which scope(s) a rule is valid, and which type of information (XSD) is concerned. A condition block describes the actual trust requirements for firing a rule, e.g., minimum personal trust, recommendation, and reputation of the requesting community member. The resulting action is a transformation of the desired information (XML) with a pre-defined XSLT script, to filter content and provide restricted views. If sharing rules collide, e.g., there is a rule for all information of a given type, and a second rule that matches the uri of the requested information, the more specific (second) rule is picked.

Sharing View Definitions. The mentioned XSLT scripts for restricting XML-based information are pre-defined by domain experts (who also define XSDs of information types), and selected by end-users when defining rules. For the exemplary paper draft schema in Listing 10.1, a matching XSLT could have the structure in Listing 10.4. After applying this script, only paper title, a contact person, categories, and keywords are visible to the requester, while the actual (co-)authors, abstract, document body, modification date, and linked resources are omitted. The output of transformations are HTML fragments that are directly embedded in a dynamic (X)HTML page and rendered in a Portlet of the Collaboration Portal.

```
<?xml version="1.0"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="html" encoding="UTF-8" indent="yes" />
   <xsl:template match="/">
      <h3>Shared Paper Draft (Restricted View)</h3>
      <xsl:apply-templates />
   </xsl:template>
   <xsl:template match="paperdraft">
      Title: <xsl:apply-templates select="title"/> <br/>
      Contact Details: <xsl:apply-templates select="contact"/> <br/>
      Categories: <xsl:for-each select="category">
                  <xsl:apply-templates/>, </xsl:for-each> <br/>
      Keywords:   <xsl:for-each select="keyword">
                  <xsl:apply-templates/>, </xsl:for-each> <br/>
   </xsl:template>
</xsl:transform>
```

Listing 10.4: Exemplary view on paper drafts.

Querying Information Collections. In contrast to the extensively discussed case of an already referenced information (identified by a well-known uri), community members will also search the network for larger sets of data (*uri/listInfos*). For instance, 'who are co-workers of member n_i?', or 'what are the documents of n_i in the dissemination scope?'.

Algorithm 11 Discover information of $type$ in the network

```
Require: information $type$, requester $requ$
$sharedInfoXML[\,] \leftarrow \emptyset$
for each $e \in$ getInfoCatalogEntries($type$) do
    if $\nexists\, jointActivity(requ, e.owner)$ then
        continue loop
    $trustRel \leftarrow$ getTrustRelation($e.owner, requ, e.scope$)
    $rule \leftarrow$ getRule($e.owner, e.type, e.mode, trustRel$)
    $view \leftarrow$ getView($rule$)
    $info \leftarrow$ getInformation($e.uri$)
    $info' \leftarrow$ applyTransformation($info, view$)
    if $info' \neq \emptyset$ then
        add($info', sharedInfoXML[\,]$)
return $sharedInfoXML[\,]$
```

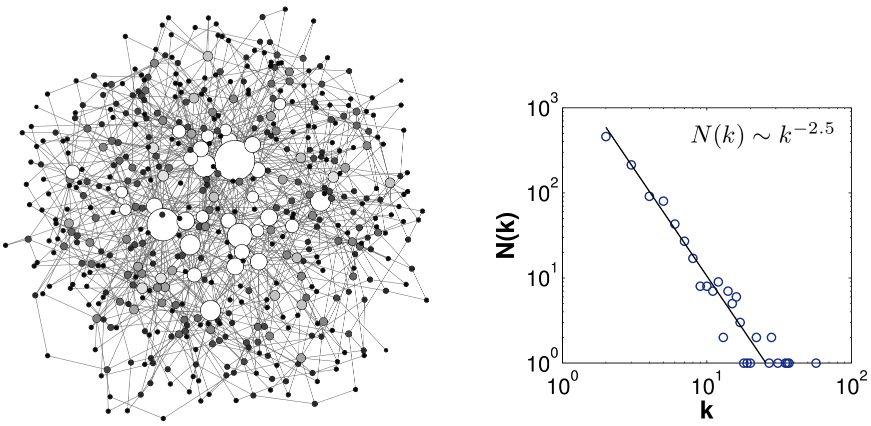
Algorithm 11 depicts the order of requests from the Sharing Proxy's perspective, when requester $requ$ queries for information of a particular $type$ in the whole network. After retrieving all catalog entries of the $type$ of interest, each entry is processed. The trust relation to the corresponding information owner in the selected scope is evaluated, and configured rules applied (using XSLTs). Finally, all found information is returned in its individually restricted shape.

10.4 Evaluation and Discussion

A fundamental aspect of our trust-based adaptation approach is the context-awareness of data and trust. Due to the high complexity of large-scale networks comprising various kinds of interactions, distinct scopes of trust, and large blocks of shared information, we evaluate the feasibility of our framework by well-directed performance studies. We focus on the most critical parts, i.e., potential bottlenecks, in our system, in particular, on (i) trust inference upon interaction logs, (ii) trust provisioning, (iii) and the overall performance of trusted information sharing. The conducted experiments address general technical problems and research challenges in complex networks, such as emerging relations in evolving structures, graph operations on large-scale networks, and information processing with respect to contextual constraints.

10.4.1 Preparation

For conducting our performance studies, we generate an artificial interaction and trust network that we would expect to emerge under realistic conditions. For that purpose we utilize the *preferential attachment model* of Barabasi and Albert to create[2] network structures that are characteristic for *science collaborations* [97].



(a) Scale-free graph structure.

(b) Power-law node distribution.

Figure 10.7: Generated network using the preferential attachment model.

[2] see JUNG graph library: `http://jung.sourceforge.net`

As shown in Figure 10.7 for a graph with 500 nodes, the output is a scale-free network with node degrees[3] following a power-law distribution. These structures are the basis for creating realistic interaction logs that are used to conduct trust inference experiments. For a graph $G = (N, E)$, we generate in total $100 \cdot |E|$ interactions between pairs of nodes (n_i, n_j). In our experiments we assume that 80% of interactions take place between 20% of the most active users (reflected by hub nodes with high degree). Generated interactions have a particular type (support request/response, activity success/failure notification) and timestamp, and occur in one of two abstract trust scopes. While we payed attention on creating a realistic amount and distribution of interactions that are closely bound to node degrees, the interaction properties themselves, i.e., type and timestamp, do not influence the actual performance study (because they do not influence the number of required operations to process the interaction logs). Furthermore, we created required XML artifacts, including some information to be shared, catalog entries, and common sharing rules (accounting for trust and recomendation) and views.

10.4.2 Experiments

For the following experiments, the Sharing Proxy and the backend services are hosted on a server with Intel Xeon 3.2GHz (quad), 10GB RAM, running Tomcat 6 with Axis2 1.4.1 on Ubuntu Linux, and MySQL 5.0 databases. The client simulation runs on a Pentium 4 with 2GB on Windows XP, and is connected with the servers through a local 100MBit Ethernet.

Interaction Logging and Trust Inference. Through utilizing available interaction properties, we calculate three metrics (i) *average response time*, (ii) *success rate* (ratio of success to the sum of success and failure notifications), and (iii) *support reciprocity* (ratio of processed to sent support requests). Individual response times are normalized to $[0, 1]$ with respect to the highest and lowest values in the whole network. Confidence $c(n_i, n_j)$ (see the trust model in Chapter 5) between each pair of connected nodes accounts for all three metrics equally. If the amount of interactions $|I(n_i, n_j)|$ between a pair (n_i, n_j) is below 10, we set the reliability of confidence to $\frac{|I(n_i,n_j)|}{10}$, else we assume a reliability of 1. Trust is calculated (for two independent scopes) by multiplying confidence $c(n_i, n_j)$ with its reliability $\rho(c(n_i, n_j))$.

We measure the required time to completely process the interaction logs, including reading logs from the interaction database (SQL), aggregating logs and calculating metrics, normalizing metrics (here only the response time, because the values of other metrics are already in $[0, 1]$), computing trust, and updates in the trust graph (EMA with $\alpha = 0.5$).

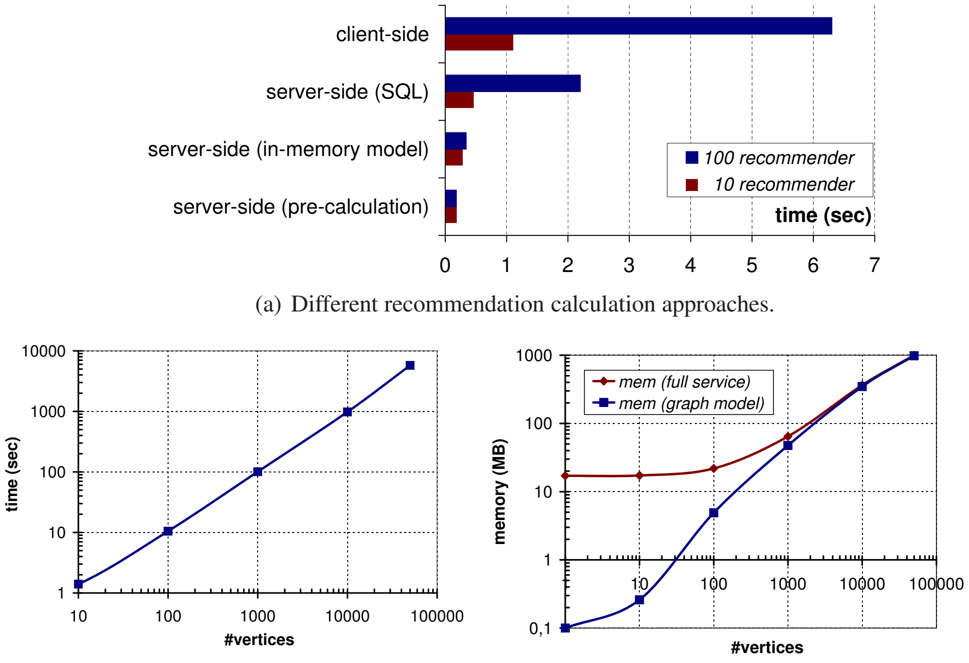
network characteristics	trust computation time
Small-scale: 100 nodes, 191 edges	1 min 44 sec
Medium-scale: 1000 nodes, 1987 edges	17 min 20 sec
Large-scale: 10000 nodes, 19983 edges	173 min 19 sec

Table 10.2: Trust inference performance results.

[3] the node size is proportional to the degree; white nodes are 'hubs'

The results show that especially for medium and large networks only a periodic offline calculation is feasible.

Network Management and Trust Provisioning. This second set of experiments, deal with trust provisioning and the calculation of recommendation and reputation on top of a large-scale trust network (10000 nodes). Figure 10.8(a) depicts the required time in seconds to calculate the recommendation $\tau^s_{rec}(n_i, n_j)$, having 10 and 100 recommender (i.e., intermediate nodes on connecting parallel paths (n_i, n_j) of length 2). Several ways to implement recommendations exist. First, a client may request all recommender nodes and their relations and calculate recommendations on the client-side. However this method is simple to implement on the provider side, it is obviously the slowest one due to large amounts of transferred data. Still retrieving all recommender and relations directly from the backend database, but performing the calculation server-side, dramatically improves the performance. However, this method produces heavy load at the provider and its database and deems not to be scalable. Therefore, we map the network data, i.e., a directed graph model with annotated nodes and edges, in memory and perform operations without the backend database. Since all data is held in memory, the performance of calculating recommendations online is comparable to provisioning of pre-calculated data only. Hence, we design our system with an in-memory graph model, and further measure some aspects of this design decision. Figure 10.8(b) illustrates required time for mapping the whole graph from the backend database to its in-memory representation. Figure 10.8(c) shows the memory consumption for instances of different sizes, first for the whole Trust Network Provider Service, and second only for the graph object itself.



(a) Different recommendation calculation approaches.

(b) Graph mapping time.

(c) Memory consumption.

Figure 10.8: Performance tests for mapping the graph model.

Overall End-To-End Performance and Caching. The overall process of trusted information sharing involves several backend services. Communicating with and retrieving data from these Web services is time-intensive, especially if they are frequently utilized and/or large amounts of data are transferred (and processed by respective SOAP stacks). Besides the actual *Information Repository*, we identified the *Information Catalog*, *Sharing View Service* and *Sharing Rule Service* as the most data-intensive services. Therefore, we studied the overall performance when caching data. In particular, the *Sharing Proxy* implements the widely adopted strategy of self-pruning cache objects [43].

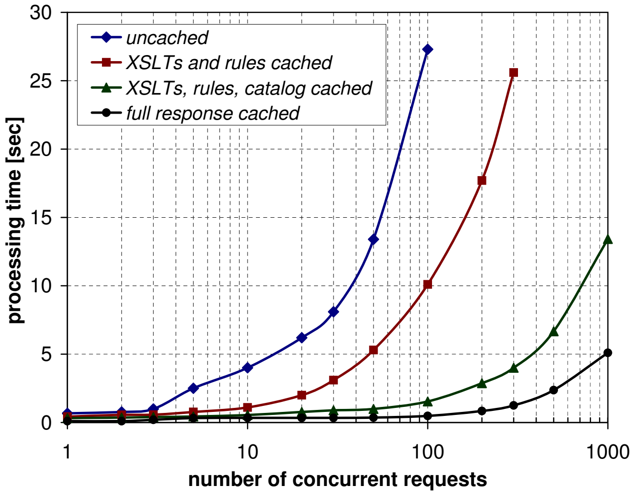


Figure 10.9: Overall performance of the Sharing Framework.

Figure 10.9 depicts the required time of the *Sharing Proxy* to process different amounts of concurrent client requests. In detail, we measured the processing time (i) without any caching mechanisms, (ii) when caching only rarely changed sharing rules and associated sharing views (XSLTs), (iii) when caching rules, XSLTs, and catalog entries, (iv) for delivering the response only, i.e., providing the already transformed and cached information. The results show that with applying different caching strategies the overall performance can be significantly increased. However, depending on the application domain, a trade-off between performance and up-to-dateness of cached data has to be carefully considered.

CHAPTER 11

Supporting Network Formation under Privacy Constraints

Outline. This chapter deals with privacy issues of utilizing data mining for trust inference and profile sharing for group formations in a Web of Social Trust.

Contents

11.1 Motivation

Small and medium-sized organizations create alliances to compete with global players, to cope with the dynamics of economy and business, and to harvest business opportunities that a single partner cannot take. In such networks where companies, communities, and individuals form virtual organizations, collaboration support is a major research track.

Individuals and companies that are interested in collaborations register at collaboration portals, where they can flexibly discover partners to form temporal alliances [17]. The collaborations in such networks usually span numerous individuals distributed over various organizations and locations. Due to the scale of these networks it is impossible for the individuals to keep track of the dynamics in such networks.

However, the recent adoption of service-oriented concepts and architectures permits the (semi-)automatic management of member profiles and network structures. In particular, SOA provides the functional means to allow loose coupling of entities through predefined interfaces and well-formed interaction messages. Upon SOA, monitoring of interactions enables the inference of social relations through mining logs. Hence, we use SOA to support and guide human interactions in collaborations by utilizing social relations.

The automatic inference and adaptation of relations between network members [90, 117] has several advantages. Negative influences, such as using outdated information, do not exist compared to manually declared relations. Moreover, monitoring of interaction behavior allows timely adaptations in ongoing collaborations, for instance, updates of member profiles based on successes in recent collaborations and collected experiences, without major user intervention.

In this chapter, we focus on supporting group formations in virtual environments by accounting for the individuals' social relations, especially *social trust*. Particularly in collaborative environments, where users are exposed higher risks than in common social network scenarios [30], and where business is at stake, considering social trust is essential to effectively guide human interactions [83].

Use case scenarios for applying *Trustworthy Group Formation* include:

- *Team Formation in Collaboration Environments*, mostly relying on recent collaboration behavior, previous successes, and member recommendations. A typical use case is the formation of a team of experts with different expertises to perform a given task.

- *Social Formation of Campaigns*, mainly focusing on people's interests and interest similarities for targeted notifications and advertisement. A typical use case is the formation of interest groups in social campaigns.

In this work, we introduce concepts and tools to facilitate the formation processes by allowing network members to browse the *Web of Social Trust*. This enables the users to discover trustworthy partners and to study their shared profile information. Hence, members initially providing more information to others are more likely to be able to set up collaborations. However, privacy of members has to be maintained. Thus, it is crucial to account for a balance between disclosing and protecting sensible profile information. Figure 11.1 underlines this required tradeoff.Finally, two major aspects have to be considered: (i) *which* profile information is shared to facilitate the set up of future collaborations, (ii) with *whom* is this information shared in order to maintain privacy.

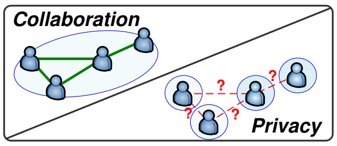


Figure 11.1: Supporting the balance between collaborations and privacy.

This chapter deals with the following contributions:

- *Privacy Patterns in Networks.* We discuss privacy-aware profile sharing patterns, applicable in a wide range of collaboration and social environments, that account for relationships between network members.

- *Collaborative SOA.* We introduce a SOA-enabled collaboration environment and discuss its advantages regarding the automatic determination of relations used to support group formation processes.
- *Privacy-aware Profile Sharing Model.* We highlight common privacy issues related to interaction mining and a model enabling members to share joint collaboration successes without violating their partners' privacy.
- *Application and Evaluation.* We introduce an end-user tool that supports network formations by exploring the *Web of Social Trust*, and discuss its applicability.

11.2 Privacy-aware Group Formations

We deal with supporting formations, e.g., composing teams and creating sub-communities, in social and collaborative environments, where the single members are connected through a *Web of Social Trust* [7]. We strongly believe that in realistic scenarios, interactions and network structures are too complex to enable entirely automatic group formation processes. There are several tradeoffs that have to be taken into account, such as considering personal relations of future group members, their expertise areas, formal skills, but also company memberships and employment status, current work load and so on. Hence, we argue the formation has to be performed by a human. However, we think supporting people in group formation processes with powerful tools will alleviate this task. Our network management approach consists of the following features: (i) *Network Member Profile Creation.* Static profiles comprise names, business areas, contact data; while dynamically adapting profiles reflect previous collaboration successes, preferences, behavior, and collected experiences. (ii) *Profile Sharing.* Profiles should be shared with network members to facilitate collaborations. However, to maintain privacy, information should be shared with trustworthy members only. (iii) *Collaboration Network Visualization.* Relations between network members emerge when performing joint activities. Recently successful compositions are visualized for reuse in future collaborations.

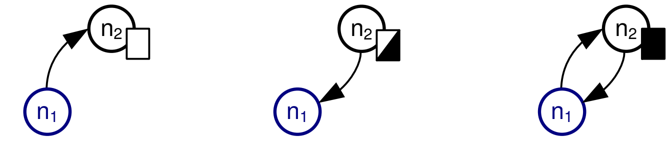
11.2.1 Formations in a Web of Social Trust

Accounting for a *Web of Social Trust* supports the discovery, selection, and group formation; and therefore the composition of network members for particular purposes. The formation process usually comprises the following three steps: (i) *Discovery of Members* by querying their static and/or dynamic profiles, including their interests and expertise, as well as the requester's preferences regarding trust and reputation. (ii) *Evaluation of Network Members* by accounting for their profiles and community embedding, e.g., a member's frequent partners. This step of the formation process discovers someone's reputation and further potential collaboration partners who can be considered in the formation process. Especially, when setting up teams, picking persons who are already familiar with each other's working style can be very beneficial. (iii) *Group Set-up* after member evaluation and selection. Depending on the environment, people are either selected after a negotiation phase, e.g., virtual team formation in business scenarios, or selected without their explicit agreement, e.g., for spreading messages in campaigns or distributing notifications based on interests and expertise areas. Our privacy approach protects users from spam and unwanted notifications,e.g., messages from untrusted members, as these notifications are directed according their periodically determined interests.

11.2.2 Privacy-aware Browsing Patterns

Allowing users to browse the *Web of Social Trust*, e.g., to study relations and previously successful compositions, raises several privacy concerns: *Who is allowed to retrieve someone's profile? Which collaboration relationships are revealed?* It is obviously unacceptable that every member of the network who performs group formations, can access all data of each other. Usually, sensible data is shared with close partners only, while general information may be shared with a larger group.

In this work, we distinguish three levels of information sharing: (i) *basic profiles* describe some fundamental properties such as name and contact details; (ii) *success stories* comprise information about previous successful activities that have been jointly performed with other members; (iii) *relations to partners*, i.e., referees, are gathered through mining of interactions in prior success stories.



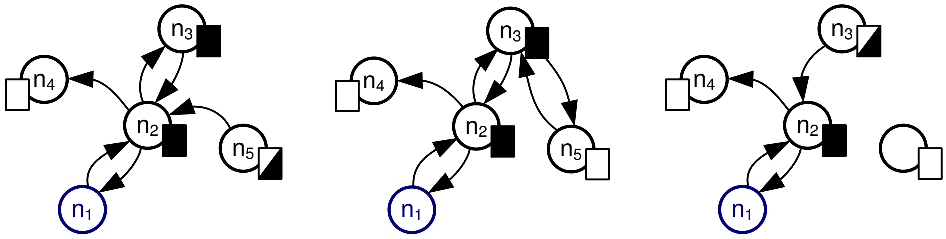
(a) Basic profile. (b) Success stories. (c) Relations.

Figure 11.2: Fundamental patterns for sharing profile data.

Figure 11.2 depicts the fundamental patterns for privacy-aware browsing of the *Web of Social Trust*. They depict mandatory relations in the *Web of Social Trust* for sharing profile information. The empty, half-filled, and full filled document symbols reflect the amount of shared profile information with the requester n_1: basic profiles, success stories, relations. Let us assume n_1 browses the *Web of Social Trust* and wants to retrieve profile information from collaboration partners. The first pattern (Figure 11.2(a)) allows him to reveal the basic profile of trusted network partners. However, n_2 only shares success stories with n_1 if n_2 trusts n_1 to some extent (Figure 11.2(b)). Relying on mutual trust in collaborations, n_1 and n_2 both share information about relations to their collaboration partners (Figure 11.2(c)).

With the fundamental patterns, only profiles, success stories, and relations from direct neighbors can be retrieved. Since this would not allow to sufficiently browse the *Web of Social Trust*, there are more advanced patterns to expand the *Circle of Trust*. Within the circle of requester n_1, members share personal data – even if they are not directly connected to n_1 – but still considering their privacy (Figure 11.3).

Propagation of Profiles (Figure 11.3(a)) allows member n_1 to browse the profile of n_3. However there is no direct connection between them, both have a transitive relation through n_2. In detail, because n_3 trusts n_2 and thus shares his profile, and n_2 trusts n_1, n_2



(a) Propagation. (b) Blocking Nodes. (c) Success Filtering.

Figure 11.3: Advanced *Web of Social Trust* browsing patterns.

shares his perspective on n_3 with n_1. This propagation mechanism can be interpreted as n_2 *recommending* n_3 to n_1 (e.g., realized with FOAF[1]) and extends n_1's *Circle of Trust*. Propagation is enabled by concatenating fundamental sharing patterns along paths with predefined lengths.

Blocking Nodes (Figure 11.3(b)) terminate the propagation of information in the *Web of Social Trust*. Profile sharing is restricted to members within a certain distance (i.e., the propagation path length). For instance, if the propagation path has a length of two hops, n_5 does not reveal success stories and relations to n_1, even though a path of mutual trust exists between them. Furthermore, it is not possible to propagate success stories or relations over a node that shares only basic profile information itself (here: n_4).

Success Filtering. (Figure 11.3(c)) means that only distinguished positive collaboration experiences are explicitly highlighted. Spreading information about unsuccessful collaborations, and low trust relations – a form of defamation – is thereby avoided. For instance, let us assume prior collaborations between n_2 and n_5 were not successful, so the identity of n_5 and its relations are hidden from n_1. Note, again there is no need for n_1 to have a personal relation to the member of interest. Users can individually configure if they like to make beneficial relations public.

11.3 Privacy in Collaborative SOA

We discuss a flexible activity-centric collaboration network using advanced SOA concepts, supporting the discovery of partners, interaction monitoring and patterns, and relation management through mining. For that purpose, we introduce a novel model that enables flexible sharing of joint collaboration successes.

11.3.1 Flexible Collaborations in SOA

During collaborations, network members interact, for instance, by exchanging documents. Collaborations in SOA means that all interactions are performed through Web services. Even the capabilities of humans are described by WSDL and communication takes place with SOAP messages (see Human-Provided Services [106] and BPEL4People[1]. In the

[1] Friend-Of-A-Friend Specification: `http://xmlns.com/foaf/spec/`

scenario depicted by Figure 11.4, the two members n_1 and n_2 perform activity a_1, n_2 and n_3 perform activities a_2 and a_3, and so on. Activities represent interaction contexts, reflected by the dashed areas, that hold information about involved actors, goals, temporal constraints, assigned resources, etc. Hence, an activity holistically describes the context of an interaction in our environment model [117].

Logged interactions are periodically analyzed and aggregated. Various metrics describe the collaboration behavior and attitude of network members, including responsiveness, availability, or reciprocity [90]. Finally, interaction metrics are interpreted according to pre-defined domain rules, and the degree of trust between each pair of previously interacting members is determined. The exact mechanism is has been studied in Chapter 4. In this chapter, we assume we are able to infer meaningful social relations and focus on privacy-awareness in formation processes.

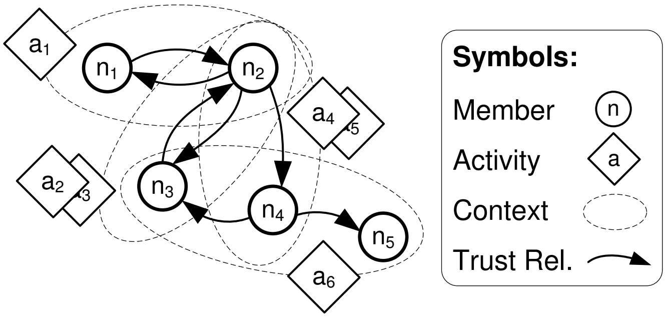


Figure 11.4: Activity-centric collaboration model.

Using interaction logging and mining in collaborative service-oriented environments enables two remarkable mechanisms:

- *Automatic Emergence of Social Relations.* Based on observed interactions, calculated behavior metrics, and the results of their interpretation, trust relations emerge between network members [117]. These relations are kept updated without the need for manual intervention.
- *Dynamic Profile Adaptations.* While network members usually manage their profiles manually, our approach allows to adapt them based on monitored collaborations; for instance, updating collected experiences and main expertise areas automatically.

11.3.2 Adaptive Profile Sharing Model

While the dynamically adapting *Web of Social Trust* represents a convenient way for network members to individually share profile data, the representation of the actually shared data is still undefined. As previously mentioned, we enable members to share basic profiles, success stories, and personal relations. For that purpose, we utilize three different models (i) the *Basic Profile Model*, (ii) the *Activity Involvement Model*, and (iii) the *Trust Graph Model*. Whenever one member requests profile information about a neighbor, s/he receives parts from respective models with regard to trust paths in the *Web of Social Trust*.

Therefore, we do not only allow members to share their own isolated profiles, but also enable the propagation of profiles along transitive relations and sharing of joint activity information.

Trust Graph Model. Let $G_\tau = (N_n, E_\tau)$ be a graph reflecting the *Web of Social Trust* with N_n denoting the set of network members and E_τ the set of directed edges connecting pairs of nodes. Each edge is associated with further metrics that are used to determine trustworthy behavior. Figure 11.4 depicts this model (ignore activity components).

Activity Involvement Model. The involvement of members in activities is modeled as bipartite graph $G_a = (N_n, N_a, E_a)$ (Figure 11.5). It comprises collaboration success stories; in our model successfully performed activities $a_i \in N_a$ and their participating members $n_j \in N_n$. An edge $e_a(n_j, a_i) \in E_a$ reflects that member n_j has been involved in activity a_i. A list of further properties may be assigned to an edge, for instance, the degree of participation or involvement role. After finishing an activity, each involved member can decide if s/he wants to share information about this collaboration as *success story*. However collaborations can be mostly categorized in terms of successful or failed from a global perspective, the individual member views may vary. Typically, newcomers will emphasize a certain activity as a success story, while experienced members categorize it as daily routine. Hence, members of finished activities can decide themselves which ones shall be explicitly included in their profiles (see dashed lines in Figure 11.5); thus, providing a personalized view on success stories. For instance, while n_2 emphasizes a_5 as success, n_4 does not. So, n_2 is not allowed to share the involvement of n_4 in that activity with n_1. According to emerged trust relations (Figure 11.4) more or less information is revealed, applying the previously introduced privacy patterns.

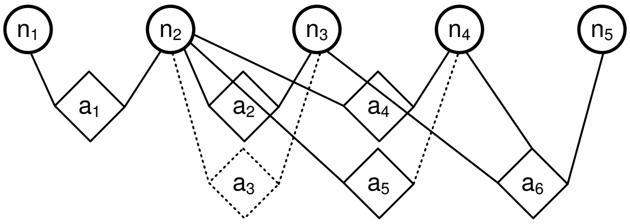
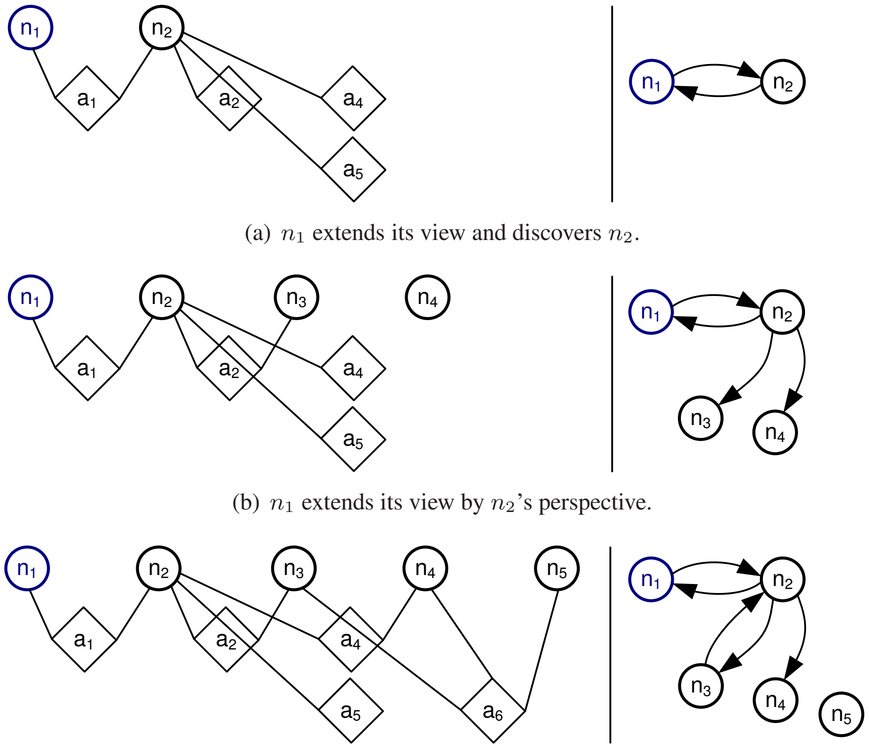


Figure 11.5: Activity-centric involvement model.

Basic Profile Model. The basic profile, attached to each node in N_n, comprises fundamental data about a member, such as name, organizational affiliations, and contact details. This basic profile is mainly static.

Shared Network Data. Finally, network members share subsets of (i) basic profiles bound to nodes in N_n, (ii) success stories reflected by G_a, and (iii) trust relations in G_τ. Figure 11.6 shows an example for data shared with n_1 when n_1 incrementally extends its view and requests data about neighbors and neighbors' neighbors. On the left side shared success stories are depicted, while on the right side shared personal relations are shown. Members share different amounts of information with n_1 through propagation according to dynamic trust G_τ for the given scenario in Figure 11.4. For instance, n_1 has no view on the trust relation from n_4 to n_5, since there is no mutual trust path from n_1 to either n_4 or n_5.



(a) n_1 extends its view and discovers n_2.

(b) n_1 extends its view by n_2's perspective.

(c) n_1 extends its view by n_3's and n_4's perspective.

Figure 11.6: Example applying browsing patterns from n_1's perspective.

11.3.3 Privacy-Aware Network Browsing Algorithm

We present Algorithm 12 that deals with sharing of basic profiles in N_n, success stories in G_a and personal relations in G_τ. The shared network segment S contains subsets of data managed by these models, and is incrementally extended. This enables a requester, i.e., the origin node n_o to browse through the *Web of Social Trust* by extending its view, i.e., S, through one of the connected nodes n_e step by step. The implementation and specific application of this approach is described in the next section.

The algorithm comprises three main parts (see comments), for adding nodes and their basic profiles, adding success stories, and personal relationships according to the previously defined browsing patterns. The functions `predec()` and `succ()` provide the predecessors and successors of a given node in a directed graph, while `neighbors()` provides connected nodes in an undirected graph. Furthermore, `isShared()` determines if a user shares a given activity as success story, and `edge()` returns the edge between two given nodes (iff one exists). Finally `addToS()` extends the shared network information segment S with provided nodes, edges, and activities. For the sake of clarity, we neglect blocking behavior when exceeding the maximum distance from n_o to n_e. Note, that the algorithm does not test, if an element – node or edge – is already included in S. The shared segment s does not contain duplicates since nodes and edges are managed in (hash) sets.

Algorithm 12 Dynamic extension of shared network data S.

```
Input: origin node $n_o$, extension node $n_e$
Global: $G_\tau = (N_n, E_\tau)$, $G_a = (N_n, N_a, E_a)$, $S = (G^s_\tau, G^s_a)$
function EXTENDVIEW($n_o, n_e$)
    /* add basic profiles of all trustors and trustees */
    $N'_n \leftarrow$ predec($n_e, G_\tau$) $\cup$ succ($n_e, G_\tau$)
    for each $n \in N'_n$ do
        addToS($n, N^s_n$)
    /* add success stories */
    $N'_n \leftarrow$ predec($n_e, G_\tau$)
    for each $n \in N'_n$ do
        $N'_a \leftarrow$ neighbors($n, G_a$)
        for each $a \in N'_a$ do
            if isShared($a, n$) then
                addToS($a, N^s_a$)
                addToS(edge($n, a, G_a$), $E^s_a$)
    /* add personal relations */
    if (predec($n_e, G^s_\tau$) $\cap$ succ($n_e, G^s_\tau$)) $\neq \emptyset \vee n_e = n_o$ then
        $N'_n \leftarrow$ predec($n_e, G_\tau$) $\cap$ succ($n_e, G_\tau$)
        for each $n \in N'_n$ do
            if $\exists$ edge($n_e, n, G_\tau$) then
                addToS(edge($n_e, n, G_\tau$), $E^s_\tau$)
```

11.4 Implementation and Application

In this section we highlight the extensions of the *VieTE* framework to support privacy-aware provisioning of profiles, relations and shared activities.

VieTE Extensions Overview. VieTE (see Chapter 4 for details) consist of three layers for (i) monitoring and logging interactions, (ii) trust inference, (iii) and trust provisioning. We address typical privacy concerns regarding logging and provisioning of data by enabling the definition of:

- *Logging Rules* that determine what interaction channels are monitored and occuring interactions logged. Of course, the more interactions are logged, the more fine-grained social relations can be inferred and thus, formations supported, e.g., based on tight trust relations.

- *Provisioning Rules* that realize the aforementioned privacy browsing patterns, and, thus, which profile information is revealed and under what circumstances.

Activity Model for Flexible Collaborations. Network members share subsets of basic profiles, joint success stories, and relations. All potentially shared data is managed by the adopted activity model depicted in Figure 11.7. Note, the entity `Involvement` does not only contain the role of a network member in an activity, but also his/her sharing preferences (`isSharedSuccess`) of a certain success story. Activities are managed through a collaboration portal, as previously discussed.

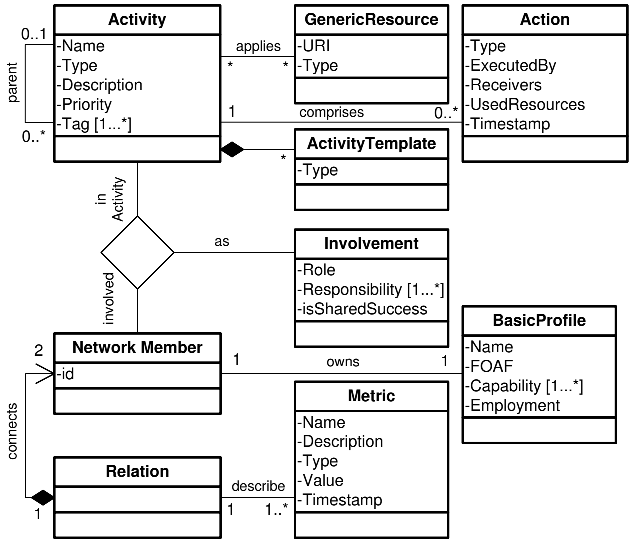


Figure 11.7: Adapted activity management and profile data model (excerpt).

Interaction Mining and Trust Emergence. Using interaction mining is an effective way to avoid sharing of static outdated data, and to unburden humans from manually updating their profiles. Member profiles, including interests and expertises, can be updated by accounting for context-aware interactions. Furthermore, interaction behavior described by various calculated interaction metrics are utilized to infer social trust relations [90, 117].

SOAP messages use WS-Security to ensure properly signed messages and strong encryption of interaction data[2]. Once interactions are stored in the framework's databases, they are used to infer higher level collaboration metrics (see Chapter 4). Therefore, low level traffic data can be purged in (short) periodic time intervals. Since we apply a centralized architecture, typical problems of P2P systems, such as propagating sensible data over several potentially untrusted nodes, can be avoided.

Adaptive Profile Sharing. Profiles, including fundamental member information, joint success stories, and personal trust relations are provided through VieTE's `Social Network Provisioning WS`. However, all requested data have to pass a `Profile Filter` that restricts a user's access to data about other members depending on social relations. For that purpose we implemented the previously discussed privacy-aware browsing patterns.

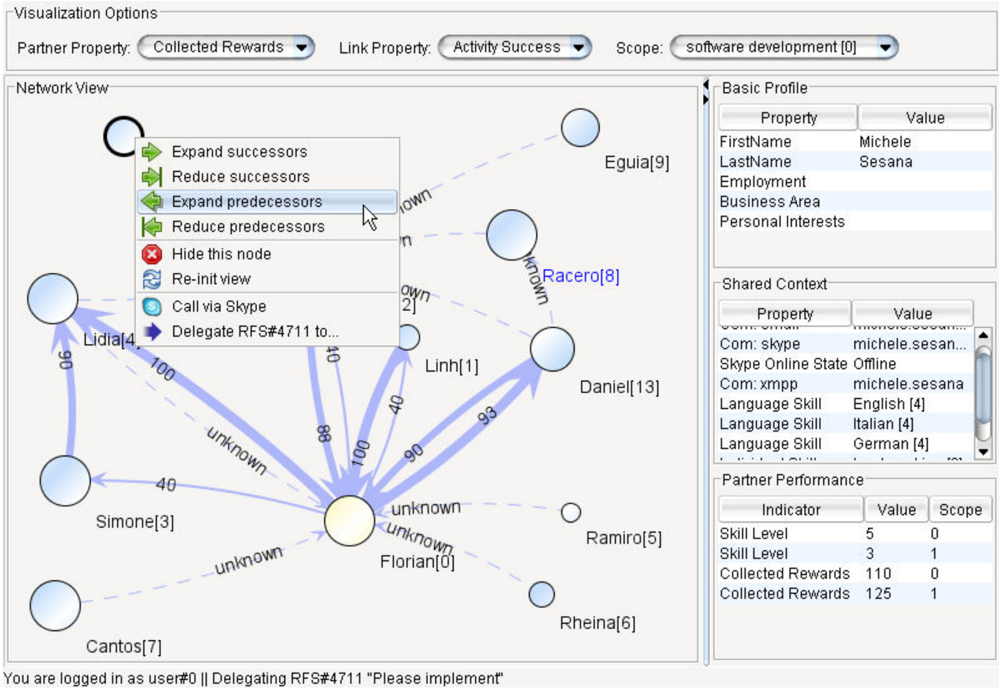
11.5 Evaluation and Discussion

11.5.1 Portal Application for Privacy-Aware Formation

We evaluated the efficiency of introduced *Web of Social Trust* browsing patterns for formations with an implementation of a Web-based *Network Browser* shown in Figure 11.8. This tool depicts the expanded network on the left side, and shared profile information on the

[2] see Apache WSS4J: `http://ws.apache.org/wss4j/`

right side. Clicking a node reveals the basic profile and some calculated metrics of a member (shown here), while clicking an edge reveals information about joint activities, where this relation emerged. Solid edges represent relations that are described by further metrics used to determine trust; see `Link Property` *Activity Success*. Dashed lines reflect trust relations that exist due to joint success stories, however, associated metrics are not shared with the tool user. The size of nodes and width of edges respectively are proportional to the visualized metrics selected by `Partner Property` and `Link Property`.



Visualization Options

Partner Property: Collected Rewards Link Property: Activity Success Scope: software development [0]

Network View

Basic Profile

Property	Value
FirstName	Michele
LastName	Sesana
Employment	
Business Area	
Personal Interests	

Shared Context

Property	Value
Com: skype	michele.sesan...
Skype Online State	Offline
Com: xmpp	michele.sesana
Language Skill	English [4]
Language Skill	Italian [4]
Language Skill	German [4]

Partner Performance

Indicator	Value	Scope
Skill Level	5	0
Skill Level	3	1
Collected Rewards	110	0
Collected Rewards	125	1

You are logged in as user#0 || Delegating RFS#4711 "Please implement"

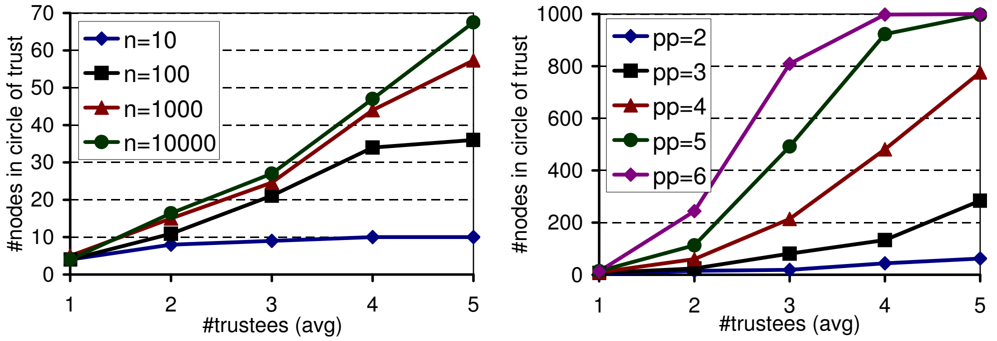
Figure 11.8: Collaboration network browser.

The formation use case starts with visualizing the user of the tool as a single (yellow) node. The user is then able to expand the network with successors and predecessors in the *Web of Social Trust* to discover potential collaboration partners in the selected `Trust Scope` (here: 'software development'). Depending on trust, partner relations and joint success stories with third parties are propagated. So, the user can incrementally discover larger segments of the *Web of Social Trust*. The tool user evaluates the members' profiles, success stories, and community embeddings, and picks single members to join a team or form a group in social campaigns. This step is supported by embedded communication tools; e.g., potential partners can be contacted with an integrated Skype client to negotiate their admission. The graph view can be conveniently zoomed to keep overview even of large-scale networks.

11.5.2 Simulations

We created artificial networks with fixed amounts of nodes and power-law distributed edges [97] to evaluate the effects of propagating profile information. Thus, we denote the complexity of a graph as the average outdegree of a node in the long tail of the degree distribution; in other words, the average number of trusted neighbors (trustees) for the bottom 90% of members. We pick random nodes from this set and run experiments for each of them until we calculate stable average results.

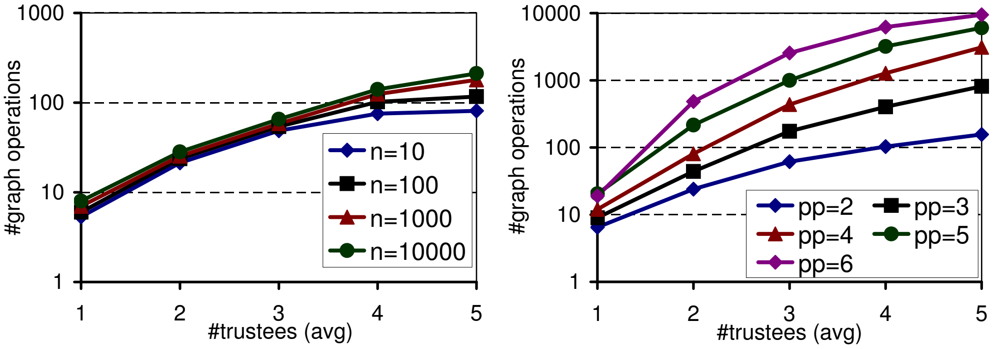
The first experiment investigates the average size of the *Circle of Trust*, depending on the number of trustees for different network sizes n and propagation path lengths pp. For that purpose we apply Algorithm 12 recursively until the whole circle (i.e., all users who share at least their joint success stories profile) is discovered. Figure 11.9 shows that for highly cross-linked graphs (i.e., $\#trustees > 2$), only short pps (max. 3 or 4 hops) are feasible. Otherwise, virtually all members are in the *Circle of Trust* - see the size of the *Circle of Trust* when propagating profile information over 5 and 6 nodes.



(a) Depending on n ($pp = 2$).

(b) Depending on pp ($n = 1000$).

Figure 11.9: Size of the circle of trust with respect to average number of trustees for different network sizes n and propagation path lengths pp.



(a) Depending on n ($pp = 2$).

(b) Depending on pp ($n = 1000$).

Figure 11.10: Required trust graph operations with respect to average number of trustees for different network sizes n and propagation path lengths pp.

A second experiment highlights the computational complexity of determining the *Circle of Trust* with Algorithm 12. While the size of the network does not considerably influence the number of required graph operations (at least for small pp), increasing pp in highly cross-linked graphs leads to exponential costs (see Figure 11.10). Graph operations include retrieving referenced nodes and edges, as well as neighbors, predecessors and successors in G_τ and G_a in VieTE's `Social Network Provisioning WS`. These operations take place on the server side before the new graph segment S is provided to the requesting client.

CHAPTER 12

Conclusion and Future Research

This work highlighted the notion of trust in service-oriented networks from various perspectives, and demonstrated its use and consequences in a wide range of application scenarios.

We strongly believe that trust is an inevitable mechanism in today's service-oriented architectures. While from a sole technical view, dealing with quality of service measurements only seems feasible, the situation in Mixed Systems fundamentally changes. When we integrate the human in the loop of service-oriented applications (such as the discussed Expert Web scenario), we also have to deal with social influences: subjective views and perceptions, motivation, incentives, risks and benefits. These influences shape the interaction behavior of people on the Web. Trust deems to be an intuitive concept to handle complex side-effects, and only trust-aware applications provide the required adaptability in mixed service-oriented environments.

Furthermore, in today's large-scale systems, a thorough technical grounding to support and automate discovery, interactions, rating, and ranking is required, since no one is able to keep track of the dynamics manually. We utilize well-established and supported Web service standards to realize our novel concepts and to ground them in state-of-the-art SOA technologies.

Future research aims at modeling complex human behavior. Currently we only capture fundamental interaction metrics, however, for complex scenarios beyond simple request-response patterns, we need approaches to model and track realistic behavior and collaborative attitudes. Furthermore, the application of our frameworks, especially the Expert Web, in real large-scale business environments would enable an evaluation of trust concepts under more realistic conditions.

Since Mixed Systems in the widest sense are already common today – however, mostly grounded in proprietary technologies – we expect promising research challenges in the future. Interactions through social networks and community platforms, recommendations and personalization on the Web, rating and ranking in the Internet of Services – trust mechanisms may be applied in all these areas to increase efficiency and effectiveness.

Bibliography

[1] A. Agrawal et al. Ws-bpel extension for people (bpel4people), version 1.0, 2007. 8, 19, 50, 110, 143

[2] Abdul-Rahman, A., and Hailes, S. Supporting trust in virtual communities. In *Hawaii International Conferences on System Sciences (HICSS)* (2000). 32

[3] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing* (1999), pp. 304–307. 8, 9

[4] Adamic, L. A., Zhang, J., Bakshy, E., and Ackerman, M. S. Knowledge sharing and yahoo answers: everyone knows something. In *International World Wide Web Conference (WWW)* (2008), ACM, pp. 665–674. 12

[5] Agichtein, E., Castillo, C., Donato, D., Gionis, A., and Mishne, G. Finding high-quality content in social media. In *ACM International Conference on Web Search and Data Mining (WSDM)* (2008), pp. 183–194. 11

[6] Altinel, M., and Franklin, M. J. Efficient filtering of xml documents for selective dissemination of information. In *International Conference on Very Large Data Bases (VLDB)* (2000), pp. 53–64. 12

[7] Artz, D., and Gil, Y. A survey of trust in computer science and the semantic web. *Journal on Web Semantics 5*, 2 (2007), 58–71. 2, 9, 17, 31, 84, 111, 141

[8] Baldauf, M., Dustdar, S., and Rosenberg, F. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing 2*, 4 (2007), 263–277. 8

[9] Balthazard, P. A., Potter, R. E., and Warren, J. Expertise, extraversion and group interaction styles as performance indicators in virtual teams: how do perceptions of it's performance get formed? *DATA BASE 35*, 1 (2004), 41–64. 8

[10] Becerra-Fernandez, I. Searching for experts on the web: A review of contemporary expertise locator systems. *ACM Transactions on Internet Technologies 6*, 4 (2006), 333–355. 113

[11] Bhatti, R., Bertino, E., and Ghafoor, A. A trust-based context-aware access control model for web-services. *Distributed and Parallel Databases 18*, 1 (2005), 83–105. 12

[12] Billhardt, H., Hermoso, R., Ossowski, S., and Centeno, R. Trust-based service provider selection in open environments. In *ACM Symposium on Applied Computing (SAC)* (2007), pp. 1375–1380. 10, 75

[13] BRADLEY, N. A., AND DUNLOP, M. D. Toward a multidisciplinary model of context to support context-aware computing. *Human-Computer Interaction 20* (2005), 403–446. 9

[14] BRANS, J., AND VINCKE, P. A preference ranking organisation method. *Management Science 31*, 6 (1985), 647–656. 12, 105, 106

[15] BRESLIN, J., PASSANT, A., AND DECKER, S. Social web applications in enterprise. *The Social Semantic Web 48* (2009), 251–267. 8

[16] BRYL, V., AND GIORGINI, P. Self-configuring socio-technical systems: Redesign at runtime. *International Transactions on Systems Science and Applications (ITSSA) 2*, 1 (2006), 31–40. 9

[17] CAMARINHA-MATOS, L. M., AND AFSARMANESH, H. Collaborative networks - value creation in a knowledge society. In *PROLAMAT* (2006), pp. 26–40. 8, 99, 139

[18] CAVERLEE, J., LIU, L., AND WEBB, S. Socialtrust: tamper-resilient trust establishment in online communities. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (2008), ACM, pp. 104–114. 10

[19] CHANG, E., HUSSAIN, F., AND DILLON, T. *Trust and Reputation for Service-Oriented Environments: Technologies For Building Business Intelligence And Consumer Confidence*. John Wiley & Sons, 2005. 17

[20] CLEAR, T., AND KASSABOVA, D. Motivational patterns in virtual team collaboration. In *Australasian Computing Education Conference (ACE)* (2005), vol. 42, pp. 51–58. 8

[21] CONNER, W., IYENGAR, A., MIKALSEN, T., ROUVELLOU, I., AND NAHRSTEDT, K. A trust management framework for service-oriented environments. In *International World Wide Web Conference (WWW)* (2009). 9

[22] COZZI, A., FARRELL, S., LAU, T., SMITH, B. A., DREWS, C., LIN, J., STACHEL, B., AND MORAN, T. P. Activity management as a web service. *IBM Systems Journal 45*, 4 (2006), 695–712. 16

[23] DEERWESTER, S., DUMAIS, S., FURNAS, G., LANDAUER, T., AND HARSHMAN, R. Indexing by latent semantic analysis. *Journal of the American society for information science 41*, 6 (1990), 391–407. 88

[24] DI NITTO, E., GHEZZI, C., METZGER, A., PAPAZOGLOU, M., AND POHL, K. A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering* (2008). 9

[25] DIAO, Y., RIZVI, S., AND FRANKLIN, M. J. Towards an internet-scale xml dissemination service. In *International Conference on Very Large Data Bases (VLDB)* (2004), pp. 612–623. 12

[26] Dias, L. C., Costa, J. P., and Climaco, J. N. A parallel implementation of the promethee method. *European Journal of Operational Research 104*, 3 (1998), 521–531. 107

[27] Dom, B., Eiron, I., Cozzi, A., and Zhang, Y. Graph-based ranking algorithms for e-mail expertise analysis. In *Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)* (2003), pp. 42–48. 11

[28] Dustdar, S. Caramba - a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases 15*, 1 (January 2004), 45–66. 8, 16

[29] Dustdar, S., and Hoffmann, T. Interaction pattern detection in process oriented information systems. *Data and Knowledge Engineering 62*, 1 (jul 2007), 138–155. 8, 32, 59

[30] Dwyer, C., Hiltz, S. R., and Passerini, K. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. In *Americas Conference on Information Systems (AMCIS)* (2007). 12, 111, 140

[31] Dybwad, B. Think twice: That facebook update could get you robbed. http://mashable.com/2009/08/27/facebook-burglary/, online, August 2009. 12

[32] Eda, T., Yoshikawa, M., and Yamamuro, M. Locally expandable allocation of folksonomy tags in a directed acyclic graph. In *International Conference on Web Information Systems Engineering (WISE)* (2008), vol. 5175, Springer, pp. 151–162. 11

[33] Feng, D., Shaw, E., Kim, J., and Hovy, E. H. Learning to detect conversation focus of threaded discussions. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)* (2006). 72

[34] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. 60, 131

[35] Figueira, J., Greco, S., and Ehrgott, M. *Multiple criteria decision analysis: state of the art surveys*. Springer, 2005. 12

[36] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, Boston, MA, USA, 1995. 59

[37] Golbeck, J. *Computing with Social Trust (Human-Computer Interaction Series)*, 1 ed. Springer, December 2008. 111

[38] Golbeck, J. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web 3*, 4 (2009), 1–33. 10, 11, 111

[39] GOLD, N., KNIGHT, C., MOHAN, A., AND MUNRO, M. Understanding service-oriented software. *IEEE Software 21*, 2 (2004), 71–77. 101

[40] GOLDER, S. A., AND HUBERMAN, B. A. The structure of collaborative tagging systems. *The Journal of Information Science* (2006). 11

[41] GOMBOTZ, R., AND DUSTDAR, S. On web services workflow mining. In *Business Process Management Workshops* (2005), pp. 216–228. 23

[42] GOMEZ, V., KALTENBRUNNER, A., AND LOPEZ, V. Statistical analysis of the social network and discussion threads in slashdot. In *International World Wide Web Conference (WWW)* (2008), pp. 645–654. 12, 78, 80

[43] GOODMAN, B. D. Accelerate your web services with caching. *IBM Advanced Internet Technology* (December 2002). 138

[44] GRANDISON, T., AND SLOMAN, M. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials 3*, 4 (2000). 9, 31, 32, 83

[45] GRIFFITHS, N. A fuzzy approach to reasoning with trust, distrust and insufficient trust. In *CIA* (2006), vol. 4149, pp. 360–374. 10, 37

[46] GUHA, R., KUMAR, R., RAGHAVAN, P., AND TOMKINS, A. Propagation of trust and distrust. In *International World Wide Web Conference (WWW)* (2004), pp. 403–412. 10, 19, 128

[47] GYÖNGYI, Z., GARCIA-MOLINA, H., AND PEDERSEN, J. Combating web spam with trustrank. In *International Conference on Very Large Data Bases (VLDB)* (2004), pp. 576–587. 11

[48] HARRISON, B. L., COZZI, A., AND MORAN, T. P. Roles and relationships for unified activity management. In *International Conference on Supporting Group Work (GROUP)* (2005), pp. 236–245. 8

[49] HAVELIWALA, T. H. Topic-sensitive pagerank. In *International World Wide Web Conference (WWW)* (2002), pp. 517–526. 11, 115

[50] HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., AND RIEDL, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 5–53. 11

[51] HESS, C., STEIN, K., AND SCHLIEDER, C. Trust-enhanced visibility for personalized document recommendations. In *ACM Symposium on Applied computing (SAC)* (2006), pp. 1865–1869. 12

[52] HEYMANN, P., AND GARCIA-MOLINA, H. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Tech. Rep. 2006-10, Computer Science Department, April 2006. 11

[53] HOFFNER, Y., LUDWIG, H., GREFEN, P. W. P. J., AND ABERER, K. Crossflow: integrating workflow management and electronic commerce. *SIGecom Exchanges 2*, 1 (2001), 1–10. 8

[54] HSU, M.-H., JU, T., YEN, C.-H., AND CHANG, C.-M. Knowledge sharing behavior in virtual communities: The relationship between trust, self-efficacy, and outcome expectations. *International Journal of Human-Computer Studies 65*, 2 (2007), 153–169. 12

[55] HUYNH, T. D., JENNINGS, N. R., AND SHADBOLT, N. R. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multiagent Systems (AAMAS) 13*, 2 (2006), 119–154. 32, 54, 75

[56] IBM. An architectural blueprint for autonomic computing. *Whitepaper* (2005). 9, 21

[57] JEH, G., AND WIDOM, J. Scaling personalized web search. In *International World Wide Web Conference (WWW)* (2003), pp. 271–279. 11

[58] JØSANG, A., ISMAIL, R., AND BOYD, C. A survey of trust and reputation systems for online service provision. *Decision Support Systems 43*, 2 (2007), 618–644. 2, 9, 17, 31, 70, 83, 84

[59] JURCZYK, P., AND AGICHTEIN, E. Discovering authorities in question answer communities by using link analysis. In *Conference on Information and Knowledge Management (CIKM)* (2007), pp. 919–922. 11

[60] KALEPU, S., KRISHNASWAMY, S., AND LOKE, S. W. Reputation = f(user ranking, compliance, verity). In *International Conference on Web Services (ICWS)* (2004), p. 200. 12

[61] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. In *International World Wide Web Conference (WWW)* (2003), pp. 640–651. 12

[62] KARAGIANNIS, T., AND VOJNOVIC, M. Email Information Flow in Large-Scale Enterprises. Tech. rep., Microsoft Research, 2008. 11

[63] KERSCHBAUM, F., HALLER, J., KARABULUT, Y., AND ROBINSON, P. Pathtrust: A trust-based reputation service for virtual organization formation. In *International Conference on Trust Management (iTrust)* (2006), pp. 193–205. 10, 12

[64] KILNER, R. Internet shopping for burglars on social networks. http://www.insurancedaily.co.uk/2009/08/28/internet-shopping-for-burglars-on-social-networks/, online, August 2009. 12

[65] KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*, 5 (1999), 604–632. 11, 12, 109, 112, 113

[66] KOVAC, D., AND TRCEK, D. Qualitative trust modeling in soa. *Journal of Systems Architecture 55*, 4 (2009), 255–263. 9, 10

[67] LEEKWIJCK, W. V., AND KERRE, E. E. Defuzzification: criteria and classification. *Fuzzy Sets and Systems 108*, 2 (1999), 159 – 178. 39

[68] LESANI, M., AND MONTAZERI, N. Fuzzy trust aggregation and personalized trust inference in virtual social networks. *Computational Intelligence 25*, 2 (2009), 51–83. 10

[69] LEYMANN, F. Workflow-based coordination and cooperation in a service world. In *CoopIS, DOA, GADA, and ODBASE* (2006), pp. 2–16. 9

[70] LOKE, S. W. Context-aware artifacts: Two development approaches. *IEEE Pervasive Computing 5*, 2 (2006), 48–53. 9

[71] M. AMEND ET AL. Web services human task (ws-humantask), version 1.0, 2007. 8

[72] MALIK, Z., AND BOUGUETTAYA, A. Reputation bootstrapping for trust establishment among web services. *IEEE Internet Computing 13*, 1 (2009), 40–47. 9

[73] MARSH, S. Information sharing is enhanced using trust models. *PerAda Magazine (Pervasive Adaptation)* (9 2008). 12

[74] MARSH, S. P. *Formalising trust as a computational concept*. PhD thesis, University of Stirling, April 1994. 9

[75] MASSA, P. A survey of trust use and modeling in real online systems, 2007. 9

[76] MASSA, P., AND AVESANI, P. Trust-aware collaborative filtering for recommender systems. In *CoopIS, DOA, ODBASE* (2004), pp. 492–508. 10

[77] MASSA, P., AND AVESANI, P. Controversial users demand local trust metrics: An experimental study on epinions.com community. In *AAAI Conference on Artificial Intelligence* (2005), pp. 121–126. 10, 19, 71, 81

[78] MATSUO, Y., AND YAMAMOTO, H. Community gravity: Measuring bidirectional effects by trust and rating on online social networks. In *International World Wide Web Conference (WWW)* (2009), pp. 751–760. 10, 85

[79] MAXIMILIEN, E. M., AND SINGH, M. P. Toward autonomic web services trust and selection. In *International Conference on Service Oriented Computing (ICSOC)* (2004), pp. 212–221. 10, 11

[80] MCKNIGHT, D. H., AND CHERVANY, N. L. The meanings of trust. Tech. rep., University of Minnesota, 1996. 9

[81] MCLURE-WASKO, M., AND FARAJ, S. Why should i share? examining social capital and knowledge contribution in electronic networks. *MIS Quarterly 29*, 1 (2005), 35–57. 10

[82] MENDLING, J., PLOESSER, K., AND STREMBECK, M. Specifying separation of duty constraints in bpel4people processes. In *Business Information Systems* (2008), pp. 273–284. 8

[83] METZGER, M. J. Privacy, trust, and disclosure: Exploring barriers to electronic commerce. *Journal on Computer-Mediated Communication 9*, 4 (2004). 12, 140

[84] MEYER, K. A. Face-to-face versus threaded discussions: The role of time and higher-order thinking. *Journal for Asynchronous Learning Networks 7*, 3 (2003), 55–65. 10

[85] MEZGÁR, I. Trust building in virtual communities. In *PRO-VE* (2009), vol. 307 of *IFIP Conference Proceedings*, Springer, pp. 393–400. 10

[86] MICHLMAYR, E., AND CAYZER, S. Learning user profiles from tagging data and leveraging them for personal(ized) information access. In *Workshop on Tagging and Metadata for Social Information Organization, WWW* (2007). 11

[87] MOODY, P., GRUEN, D., MULLER, M. J., TANG, J. C., AND MORAN, T. P. Business activity patterns: A new model for collaborative business applications. *IBM Systems Journal 45*, 4 (2006), 683–694. 8, 16

[88] MORAN, T. P., COZZI, A., AND FARRELL, S. P. Unified activity management: Supporting people in e-business. *Communications of the ACM 48*, 12 (2005), 67–70. 8

[89] MORI, J., SUGIYAMA, T., AND MATSUO, Y. Real-world oriented information sharing using social networks. In *GROUP* (2005), pp. 81–84. 12

[90] MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. A computational model of trust and reputation for e-businesses. In *Hawaii International Conferences on System Sciences (HICSS)* (2002), p. 188. 9, 10, 32, 140, 144, 148

[91] NONNECKE, B., PREECE, J., AND ANDREWS, D. What lurkers and posters think of each other. In *Hawaii International Conferences on System Sciences (HICSS)* (2004). 10, 72

[92] OASIS. Business process execution language for web services, version 2.0, 2007. 49, 109

[93] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford University, 1998. 11, 12, 55

[94] PANTELI, N., AND DAVISON, R. The role of subgroups in the communication patterns of global virtual teams. *IEEE Transactions on Professional Communication 48*, 2 (2005), 191–200. 8

[95] Papazoglou, M. *Web Services: Principles and Technology*, 1 ed. Prentice Hall, 2007. 3

[96] Rajbhandari, S., Rana, O. F., and Wootten, I. A fuzzy model for calculating workflow trust using provenance data. In *ACM Mardi Gras Conference* (2008), p. 10. 10, 37

[97] Reka, A., and Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys. 74* (June 2002), 47–97. 119, 126, 135, 149

[98] Rheingold, H. *The Virtual Community: Homesteading on the electronic frontier, revised edition*. The MIT Press, November 2000. 10

[99] Romesburg, H. C. *Cluster Analysis for Researchers*. Krieger Pub. Co., 2004. 11

[100] Ruohomaa, S., and Kutvonen, L. Trust management survey. In *International Conference on Trust Management (iTrust)* (2005), Springer, pp. 77–92. 83

[101] Russell, N., and Aalst, W. M. P. V. D. Evaluation of the bpel4people and ws-humantask extensions to ws-bpel 2.0 using the workflow resource patterns. Tech. rep., BPM Center Brisbane/Eindhoven, 2007. 8

[102] Salehie, M., and Tahvildari, L. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems 4*, 2 (May 2009). 2

[103] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24*, 5 (1988), 513–523. 11, 88

[104] Schall, D. *Human Interactions in Mixed Systems - Architecture, Protocols, and Algorithms*. PhD thesis, Vienna University of Technology, 2009. 3, 4, 62

[105] Schall, D., Skopik, F., and Dustdar, S. Trust-based discovery and interactions in expert networks. Tech. rep., Vienna University of Technology, TUV-1841-2010-01, 2010. 6

[106] Schall, D., Truong, H.-L., and Dustdar, S. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing 12*, 3 (2008), 62–68. 1, 3, 8, 15, 19, 33, 35, 50, 104, 110, 119, 125, 143

[107] Sharon Paradesi, P. D., and Swaika, S. Integrating behavioral trust in web service compositions. In *International Conference on Web Services (ICWS)* (2009). 10

[108] Shepitsen, A., Gemmell, J., Mobasher, B., and Burke, R. Personalized recommendation in social tagging systems using hierarchical clustering. In *International Conference on Recommender Systems (RecSys)* (2008), pp. 259–266. 11, 86

[109] SHERCHAN, W., LOKE, S. W., AND KRISHNASWAMY, S. A fuzzy model for reasoning about reputation in web services. In *ACM Symposium on Applied Computing (SAC)* (2006), pp. 1886–1892. 10, 37

[110] SHETTY, J., AND ADIBI, J. Discovering important nodes through graph entropy the case of enron email database. In *Workshop on Link Analysis at the International Conference on Knowledge Discovery and Data Mining (LinkKDD)* (2005), pp. 74–81. 11

[111] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. The cycle of trust in mixed service-oriented systems. In *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2009), pp. 72–79. 2, 5, 32, 33, 34, 35, 61, 84, 123, 125, 126

[112] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. Start trusting strangers? bootstrapping and prediction of trust. In *International Conference on Web Information Systems Engineering (WISE)* (2009), pp. 275–289. 6, 10

[113] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. Modeling and mining of dynamic trust in complex service-oriented systems. *Information Systems* (2010). 5, 37

[114] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. Supporting network formation through mining under privacy constraints. In *International Symposium on Applications and the Internet (SAINT)* (2010). 6

[115] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. Trust-based adaptation in complex service-oriented systems. In *International Conference on Engineering of Complex Computer Systems (ICECCS)* (2010), pp. 31–40. 6

[116] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. Trusted interaction patterns in large-scale enterprise service networks. In *Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP)* (2010), pp. 367–374. 5, 12, 100, 104, 117

[117] SKOPIK, F., SCHALL, D., AND DUSTDAR, S. Trustworthy interaction balancing in mixed service-oriented systems. In *ACM Symposium on Applied Computing (SAC)* (2010), pp. 801–808. 5, 10, 119, 140, 144, 148

[118] SKOPIK, F., SCHALL, D., DUSTDAR, S., AND SESANA, M. Context-aware interaction models in cross-organizational processes. In *International Conference on Internet and Web Applications and Services (ICIW)* (2010). 6

[119] SKOPIK, F., TRUONG, H.-L., AND DUSTDAR, S. Trust and reputation mining in professional virtual communities. In *International Conference on Web Engineering (ICWE)* (2009), pp. 76–90. 5, 84

[120] SKOPIK, F., TRUONG, H.-L., AND DUSTDAR, S. VieTE - enabling trust emergence in service-oriented collaborative environments. In *International Conference on Web Information Systems and Technologies (WEBIST)* (2009), pp. 471–478. 16, 71

[121] Srivatsa, M., Xiong, L., and Liu, L. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *International World Wide Web Conference (WWW)* (2005), pp. 422–431. 10, 53

[122] The Economist. The role of trust in business collaboration. *An Economist Intelligence Unit briefing paper sponsored by Cisco Systems* (2008). 10, 70

[123] Theodorakopoulos, G., and Baras, J. S. On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas in Communications 24*, 2 (2006), 318–328. 10

[124] Uddin, M. G., Zulkernine, M., and Ahamed, S. I. Collaboration through computation. *Service Oriented Computing and Applications (SOCA) 3*, 1 (2009), 47–63. 10

[125] Walter, F. E., Battiston, S., and Schweitzer, F. Personalised and dynamic trust in social networks. In *ACM Conference on Recommender Systems (RecSys)* (2009), pp. 197–204. 12

[126] Wanas, N. M., El-Saban, M., Ashour, H., and Ammar, W. Automatic scoring of online discussion posts. In *Workshop on Information Credibility on the Web (WICOW)* (2008), ACM, pp. 19–26. 72

[127] Wang, Y., and Singh, M. P. Trust representation and aggregation in a distributed agent system. In *AAAI Conference on Artificial Intelligence* (2006). 10

[128] Watts, D. J. *Six degrees: The science of a connected age*. WW Norton & Company, 2003. 58, 117

[129] Welser, H. T., Gleave, E., Fisher, D., and Smith, M. Visualizing the signatures of social roles in online discussion groups. *Journal of Social Structure 8* (2007). 10

[130] Yang, J., Adamic, L., and Ackerman, M. Competing to share expertise: the taskcn knowledge sharing community. In *International Conference on Weblogs and Social Media* (2008). 11

[131] Zadeh, L. A. Fuzzy sets. *Information and Control 8* (1965), 338–353. 37, 38

[132] Zhang, J., Ackerman, M. S., and Adamic, L. Expertise networks in online communities: structure and algorithms. In *International World Wide Web Conference (WWW)* (2007), pp. 221–230. 11

[133] Zhang, J., and Figueiredo, R. J. Autonomic feature selection for application classification. In *International Conference on Autonomic Computing (ICAC)* (2006), IEEE, pp. 43–52. 9

[134] Ziegler, C.-N., and Golbeck, J. Investigating interactions of trust and interest similarity. *Decision Support Systems 43*, 2 (2007), 460–475. 10, 11, 85, 111, 128

[135] Ziegler, C.-N., and Lausen, G. Propagation models for trust and distrust in social networks. *Information Systems Frontiers 7*, 4-5 (2005), 337–358. 10

[136] Zimmermann, H.-J. *Fuzzy Set Theory and Its Applications*, third ed. Kluwer Academic Publishers, 1996. 37

[137] Zuo, Y., and Panda, B. Component based trust management in the context of a virtual organization. In *ACM Symposium on Applied Computing (SAC)* (2005), pp. 1582–1588. 10, 12

Appendix A
Collaboration Network Provider Service Specification

The *Collaboration Network Provider Service (CNP)* is the basis to develop trust-aware applications, such as presented *Trusted Information Sharing*, *Trusted Online Help and Support* through the Expert Web, and *Trustworthy Group Formation* support. Thus, CNP is the interface that connects trust inference mechanisms and the actual applications on top. Since this service is a central component of all presented software frameworks in this dissertation, we outline its implementation and interface in this appendix.

A.1 Overview of the Service

The CNP is a SOAP-based Web Service with an interface described by WSDL. The service provides data describing the *Web of Trust*, including trust relations between actors of the Mixed Systems, and their nature, i.e., underlying interaction metrics, and context of relations. In order to use this service, it is either (i) filled with synthetic data, obtained from simulations (see Chapter 4, Chapter 9, and Chapter 10); or (ii) (partly) filled with existing data sets; for instance Slashdot discussion relations and citeulike interest profiles (see Chapter 6 and Capter 7).

The service is implemented in Java using the Apache Axis2[1] Web Services Stack, and uses a MySQL[2] database backend.

A.2 Data Representation

All entities are identified by URIs, which are combined of various subparts: (i) a basepath (i.e., `http://www.infosys.tuwien.ac.at/coin`), (ii) the entitiy type (e.g., `node`) and (iii) an integer `id`. An example for a full uri that identifies a certain node is `http://www.infosys.tuwien.ac.at/coin/node#19`.

The Web service deals with the following fundamental types of entities. Their details and relations are described by the extended entitiy relationsship model (EER) in Figure A.1.:

- *Node:* A node describes either a human, service or HPS.
- *Edge:* An Edge reflects the directed relation between two nodes.

[1] `http://ws.apache.org/axis2/`

[2] `http://www.mysql.com/`

- *Metric:* Metrics describe properties of either nodes (such as the number of interactions with any service, or the number of activities involved) or edges (such as the number of invocations from a particular service by a particular human). Collaboration metrics are calculated from interactions in predefined scopes. Trust metrics are determined by interpreting Collaboration Metrics.

- *Scope:* Rules determine which interactions are used to calculate metrics (e.g., only interactions of a particular type are considered in the trust determination process; see Chapter 3). Furthermore, they are used to interpret collaboration metrics; see Chapter 4. These rules describe the constraints for the validity of metric calculations, i.e., the scope of their application. Currently common scopes are pre-configured and can be selected through a Web service interface, however they cannot be modified.

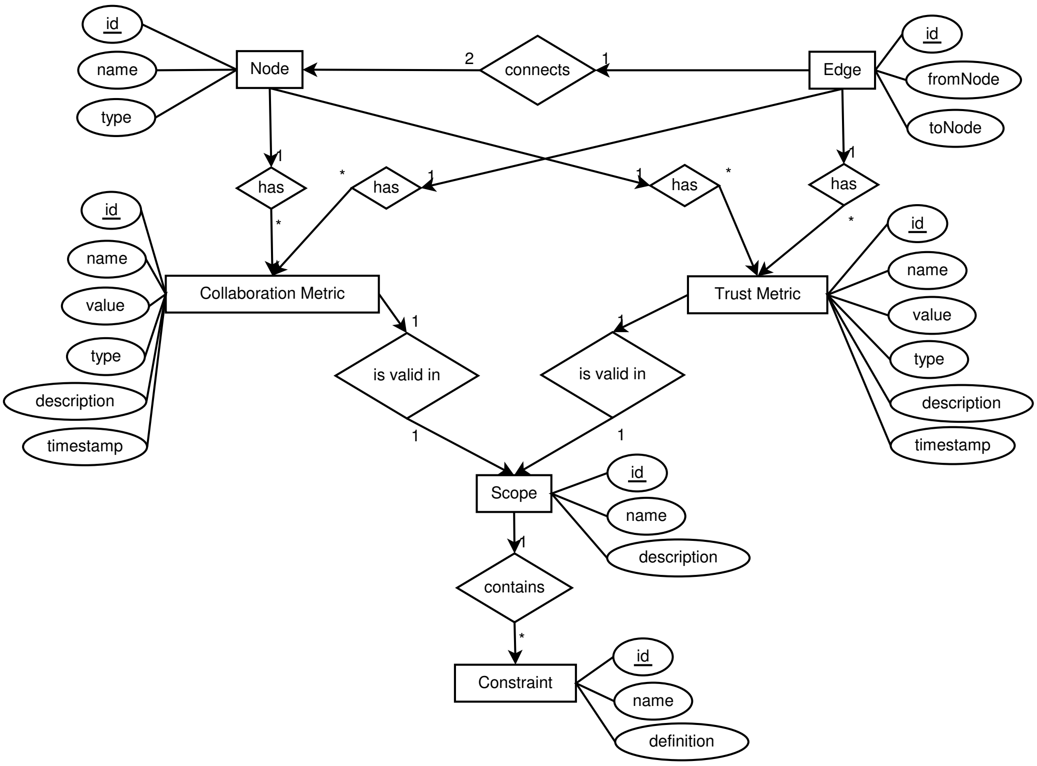


Figure A.1: Data model of the Collaboration Network Provider.

A.3 Interface Description

The Collaboration Network Provider enables the successive retrieval of the collaboration graph (*Web of Trust*) starting with a predefined node, e.g., reflecting the service user. We specify its interface as shown in Table A.1. Note, for data retrieval, metrics are merged in the entities node and edge.

operation name	parameters	description
getNode	nodeURI	get the node object with the given URI
searchNodesByName	(part of) nodeName	get a list of nodes with matching names
getAllNodes	–	get all nodes (can be restricted to a maximum number due to performance reasons)
getEdge	edgeURI	get the specified edge
getEdges	fromNodeURI, toNodeURI	get all directed edges from sourceNode to sinkNode
getOutEdges	nodeURI	get all out edges of the specified node
getInEdges	nodeURI	get all in edges of the specified node
getScope	scopeURI	get one particular scope in the network
getAllScopes	–	get all available scopes in the network
getSourceNode	edgeURI	get the node object which is the source of the given edge
getSinkNode	edgeURI	get the node object which is the sink of the given edge
getNeighbours	nodeURI, numHops	get neighbors (independent of edge orientation); the optional parameter numHops may set the maximum number of nodes between the specified node and provided nodes
getSuccessors	nodeURI	get successors of specified node
getPredecessors	nodeURI	get direct predecessors of specified node
getVersion	–	get version string

Table A.1: Collaboration Network Provider operations.

Listing A.1 shows the WSDL of this service. Note, that large parts of the description, such as the alternative SOAP 1.1. and HTTP bindings, have been omitted due to the sake of brevity.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  xmlns:ns0="http://model.collabnwprovider.coin.infosys.tuwien.ac.at/xsd"
                  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:ns1="http://collabnwprovider.coin.infosys.tuwien.ac.at"
                  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  targetNamespace="http://collabnwprovider.coin.infosys.tuwien.ac.at">
    <wsdl:documentation>CollaborationNetworkProvider</wsdl:documentation>
    <wsdl:types>
        <xs:schema xmlns:ax22="http://model.collabnwprovider.coin.infosys.tuwien.ac.at/xsd"
                   attributeFormDefault="qualified" elementFormDefault="qualified"
                   targetNamespace="http://model.collabnwprovider.coin.infosys.tuwien.ac.at/xsd">
            <xs:complexType name="CoinURI">
                <xs:sequence>
                    <xs:element minOccurs="0" name="entityType" nillable="true" type="xs:string"/>
                    <xs:element minOccurs="0" name="id" type="xs:int"/>
                    <xs:element minOccurs="0" name="path" nillable="true" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="Edge">
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="collaborationMetrics"
                                nillable="true" type="ax22:CollaborationMetric"/>
                    <xs:element minOccurs="0" name="edgeURI" nillable="true" type="ax22:CoinURI"/>
            <xs:element minOccurs="0" name="sinkNodeURI" nillable="true" type="ax22:CoinURI"/>
            <xs:element minOccurs="0" name="sourceNodeURI" nillable="true" type="ax22:CoinURI"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="trustMetrics"
                        nillable="true" type="ax22:TrustMetric"/>
            <xs:element minOccurs="0" name="updatedAt" nillable="true" type="xs:dateTime"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Metric">
        <xs:sequence>
            <xs:element minOccurs="0" name="description" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="metricURI" nillable="true" type="ax22:CoinURI"/>
            <xs:element minOccurs="0" name="name" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="scope" nillable="true" type="ax22:CoinURI"/>
            <xs:element minOccurs="0" name="type" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="updatedAt" nillable="true" type="xs:dateTime"/>
            <xs:element minOccurs="0" name="value" nillable="true" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CollaborationMetric">
        <xs:complexContent>
            <xs:extension base="ax22:Metric">
                <xs:sequence/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TrustMetric">
        <xs:complexContent>
            <xs:extension base="ax22:Metric">
                <xs:sequence/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="Node">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="collaborationMetrics"
                        nillable="true" type="ax22:CollaborationMetric"/>
            <xs:element minOccurs="0" name="name" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="nodeURI" nillable="true" type="ax22:CoinURI"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="trustMetrics"
                        nillable="true" type="ax22:TrustMetric"/>
            <xs:element minOccurs="0" name="type" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="updatedAt" nillable="true" type="xs:dateTime"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Scope">
        <xs:sequence>
            <xs:element minOccurs="0" name="description" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="name" nillable="true" type="xs:string"/>
            <xs:element minOccurs="0" name="scopeURI" nillable="true" type="ax22:CoinURI"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="tags" nillable="true" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
<xs:schema xmlns:ns="http://collabnwprovider.coin.infosys.tuwien.ac.at" attributeFormDefault="qualified"
           elementFormDefault="qualified" targetNamespace="http://collabnwprovider.coin.infosys.tuwien.ac.at">
    <xs:complexType name="Exception">
        <xs:sequence>
            <xs:element minOccurs="0" name="Exception" nillable="true" type="xs:anyType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="getEdge">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="edgeURI" nillable="true" type="ns0:CoinURI"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="getEdgeResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="return" nillable="true" type="ns0:Edge"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="getEdges">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="sourceNodeURI" nillable="true" type="ns0:CoinURI"/>
                <xs:element minOccurs="0" name="sinkNodeURI" nillable="true" type="ns0:CoinURI"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="getEdgesResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Edge"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="getInEdges">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="sinkNodeURI" nillable="true" type="ns0:CoinURI"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="getInEdgesResponse">
        <xs:complexType>
```

```
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Edge"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getOutEdges">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="sourceNodeURI" nillable="true" type="ns0:CoinURI"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getOutEdgesResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Edge"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getNode">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="nodeURI" nillable="true" type="ns0:CoinURI"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getNodeResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getSinkNode">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="edgeURI" nillable="true" type="ns0:CoinURI"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getSinkNodeResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getSourceNode">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="edgeURI" nillable="true" type="ns0:CoinURI"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getSourceNodeResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getAllNodesResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getNeighbours">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="nodeURI" nillable="true" type="ns0:CoinURI"/>
                    <xs:element minOccurs="0" name="numHops" type="xs:int"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getNeighboursResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getPredecessors">
            <xs:complexType>
                <xs:sequence>
                    <xs:element minOccurs="0" name="nodeURI" nillable="true" type="ns0:CoinURI"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getPredecessorsResponse">
            <xs:complexType>
                <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="getSuccessors">
```

```
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="nodeURI" nillable="true" type="ns0:CoinURI"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getSuccessorsResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="searchNodesByName">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="name" nillable="true" type="xs:string"/>
                        <xs:element minOccurs="0" name="isFuzzySearch" type="xs:boolean"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="searchNodesByNameResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Node"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getScope">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="scopeURI" nillable="true" type="ns0:CoinURI"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getScopeResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="return" nillable="true" type="ns0:Scope"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getAllScopesResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true" type="ns0:Scope"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getVersionResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="return" nillable="true" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="getSourceNodeRequest">
        <wsdl:part name="parameters" element="ns1:getSourceNode"/>
    </wsdl:message>
    <wsdl:message name="getSourceNodeResponse">
        <wsdl:part name="parameters" element="ns1:getSourceNodeResponse"/>
    </wsdl:message>
    <wsdl:message name="getPredecessorsRequest">
        <wsdl:part name="parameters" element="ns1:getPredecessors"/>
    </wsdl:message>
    <wsdl:message name="getPredecessorsResponse">
        <wsdl:part name="parameters" element="ns1:getPredecessorsResponse"/>
    </wsdl:message>
    <wsdl:message name="getNodeRequest">
        <wsdl:part name="parameters" element="ns1:getNode"/>
    </wsdl:message>
    <wsdl:message name="getNodeResponse">
        <wsdl:part name="parameters" element="ns1:getNodeResponse"/>
    </wsdl:message>
    <wsdl:message name="getEdgeRequest">
        <wsdl:part name="parameters" element="ns1:getEdge"/>
    </wsdl:message>
    <wsdl:message name="getEdgeResponse">
        <wsdl:part name="parameters" element="ns1:getEdgeResponse"/>
    </wsdl:message>
    <wsdl:message name="getEdgesRequest">
        <wsdl:part name="parameters" element="ns1:getEdges"/>
    </wsdl:message>
    <wsdl:message name="getEdgesResponse">
        <wsdl:part name="parameters" element="ns1:getEdgesResponse"/>
    </wsdl:message>
    <wsdl:message name="getSinkNodeRequest">
        <wsdl:part name="parameters" element="ns1:getSinkNode"/>
    </wsdl:message>
    <wsdl:message name="getSinkNodeResponse">
        <wsdl:part name="parameters" element="ns1:getSinkNodeResponse"/>
    </wsdl:message>
    <wsdl:message name="searchNodesByNameRequest">
        <wsdl:part name="parameters" element="ns1:searchNodesByName"/>
    </wsdl:message>
    <wsdl:message name="searchNodesByNameResponse">
```

```
        <wsdl:part name="parameters" element="ns1:searchNodesByNameResponse"/>
    </wsdl:message>
    <wsdl:message name="getSuccessorsRequest">
        <wsdl:part name="parameters" element="ns1:getSuccessors"/>
    </wsdl:message>
    <wsdl:message name="getSuccessorsResponse">
        <wsdl:part name="parameters" element="ns1:getSuccessorsResponse"/>
    </wsdl:message>
    <wsdl:message name="getOutEdgesRequest">
        <wsdl:part name="parameters" element="ns1:getOutEdges"/>
    </wsdl:message>
    <wsdl:message name="getOutEdgesResponse">
        <wsdl:part name="parameters" element="ns1:getOutEdgesResponse"/>
    </wsdl:message>
    <wsdl:message name="getNeighboursRequest">
        <wsdl:part name="parameters" element="ns1:getNeighbours"/>
    </wsdl:message>
    <wsdl:message name="getNeighboursResponse">
        <wsdl:part name="parameters" element="ns1:getNeighboursResponse"/>
    </wsdl:message>
    <wsdl:message name="getVersionRequest"/>
    <wsdl:message name="getVersionResponse">
        <wsdl:part name="parameters" element="ns1:getVersionResponse"/>
    </wsdl:message>
    <wsdl:message name="getAllScopesRequest"/>
    <wsdl:message name="getAllScopesResponse">
        <wsdl:part name="parameters" element="ns1:getAllScopesResponse"/>
    </wsdl:message>
    <wsdl:message name="getInEdgesRequest">
        <wsdl:part name="parameters" element="ns1:getInEdges"/>
    </wsdl:message>
    <wsdl:message name="getInEdgesResponse">
        <wsdl:part name="parameters" element="ns1:getInEdgesResponse"/>
    </wsdl:message>
    <wsdl:message name="getAllNodesRequest"/>
    <wsdl:message name="getAllNodesResponse">
        <wsdl:part name="parameters" element="ns1:getAllNodesResponse"/>
    </wsdl:message>
    <wsdl:message name="getScopeRequest">
        <wsdl:part name="parameters" element="ns1:getScope"/>
    </wsdl:message>
    <wsdl:message name="getScopeResponse">
        <wsdl:part name="parameters" element="ns1:getScopeResponse"/>
    </wsdl:message>
    <wsdl:portType name="CollaborationNetworkProviderPortType">
        <wsdl:operation name="getSourceNode">
            <wsdl:input message="ns1:getSourceNodeRequest" wsaw:Action="urn:getSourceNode"/>
            <wsdl:output message="ns1:getSourceNodeResponse" wsaw:Action="urn:getSourceNodeResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getPredecessors">
            <wsdl:input message="ns1:getPredecessorsRequest" wsaw:Action="urn:getPredecessors"/>
            <wsdl:output message="ns1:getPredecessorsResponse" wsaw:Action="urn:getPredecessorsResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getNode">
            <wsdl:input message="ns1:getNodeRequest" wsaw:Action="urn:getNode"/>
            <wsdl:output message="ns1:getNodeResponse" wsaw:Action="urn:getNodeResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getEdge">
            <wsdl:input message="ns1:getEdgeRequest" wsaw:Action="urn:getEdge"/>
            <wsdl:output message="ns1:getEdgeResponse" wsaw:Action="urn:getEdgeResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getEdges">
            <wsdl:input message="ns1:getEdgesRequest" wsaw:Action="urn:getEdges"/>
            <wsdl:output message="ns1:getEdgesResponse" wsaw:Action="urn:getEdgesResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getSinkNode">
            <wsdl:input message="ns1:getSinkNodeRequest" wsaw:Action="urn:getSinkNode"/>
            <wsdl:output message="ns1:getSinkNodeResponse" wsaw:Action="urn:getSinkNodeResponse"/>
        </wsdl:operation>
        <wsdl:operation name="searchNodesByName">
            <wsdl:input message="ns1:searchNodesByNameRequest" wsaw:Action="urn:searchNodesByName"/>
            <wsdl:output message="ns1:searchNodesByNameResponse" wsaw:Action="urn:searchNodesByNameResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getSuccessors">
            <wsdl:input message="ns1:getSuccessorsRequest" wsaw:Action="urn:getSuccessors"/>
            <wsdl:output message="ns1:getSuccessorsResponse" wsaw:Action="urn:getSuccessorsResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getOutEdges">
            <wsdl:input message="ns1:getOutEdgesRequest" wsaw:Action="urn:getOutEdges"/>
            <wsdl:output message="ns1:getOutEdgesResponse" wsaw:Action="urn:getOutEdgesResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getNeighbours">
            <wsdl:input message="ns1:getNeighboursRequest" wsaw:Action="urn:getNeighbours"/>
            <wsdl:output message="ns1:getNeighboursResponse" wsaw:Action="urn:getNeighboursResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getVersion">
            <wsdl:input message="ns1:getVersionRequest" wsaw:Action="urn:getVersion"/>
            <wsdl:output message="ns1:getVersionResponse" wsaw:Action="urn:getVersionResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getAllScopes">
            <wsdl:input message="ns1:getAllScopesRequest" wsaw:Action="urn:getAllScopes"/>
            <wsdl:output message="ns1:getAllScopesResponse" wsaw:Action="urn:getAllScopesResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getInEdges">
            <wsdl:input message="ns1:getInEdgesRequest" wsaw:Action="urn:getInEdges"/>
            <wsdl:output message="ns1:getInEdgesResponse" wsaw:Action="urn:getInEdgesResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getAllNodes">
```

```
            <wsdl:input message="ns1:getAllNodesRequest" wsaw:Action="urn:getAllNodes"/>
            <wsdl:output message="ns1:getAllNodesResponse" wsaw:Action="urn:getAllNodesResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getScope">
            <wsdl:input message="ns1:getScopeRequest" wsaw:Action="urn:getScope"/>
            <wsdl:output message="ns1:getScopeResponse" wsaw:Action="urn:getScopeResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="CollaborationNetworkProviderSOAPBinding" type="ns1:CollaborationNetworkProviderPortType">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
        <wsdl:operation name="getSourceNode">
            <soap12:operation soapAction="urn:getSourceNode" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getPredecessors">
            <soap12:operation soapAction="urn:getPredecessors" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getNode">
            <soap12:operation soapAction="urn:getNode" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getEdge">
            <soap12:operation soapAction="urn:getEdge" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getEdges">
            <soap12:operation soapAction="urn:getEdges" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getSinkNode">
            <soap12:operation soapAction="urn:getSinkNode" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="searchNodesByName">
            <soap12:operation soapAction="urn:searchNodesByName" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getSuccessors">
            <soap12:operation soapAction="urn:getSuccessors" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getOutEdges">
            <soap12:operation soapAction="urn:getOutEdges" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getNeighbours">
            <soap12:operation soapAction="urn:getNeighbours" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getVersion">
            <soap12:operation soapAction="urn:getVersion" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getAllScopes">
            <soap12:operation soapAction="urn:getAllScopes" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getInEdges">
            <soap12:operation soapAction="urn:getInEdges" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getAllNodes">
            <soap12:operation soapAction="urn:getAllNodes" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getScope">
            <soap12:operation soapAction="urn:getScope" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CollaborationNetworkProvider">
       <wsdl:port name="CollaborationNetworkProviderSOAPport_http" binding="ns1:CollaborationNetworkProviderSOAPBinding">
           <soap:address location="http://madrid.vitalab.tuwien.ac.at:8152/axis2/services/CollaborationNetworkProvider"/>
       </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Listing A.1: Collaboration Network Provider (CNP) WSDL