Institut für Informatik der Technischen Universität München

Lehrstuhl für Informatik XI

Social Relationship Management in Internet-based Communication and Shared Information Spaces

Michael Galla

Institut für Informatik der Technischen Universität München

Lehrstuhl für Informatik XI

Social Relationship Management in Internet-based Communication and Shared Information Spaces

Michael Galla

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Rudolf Bayer, Ph.D.
Prüfer der Dissertation:	1. Univ.-Prof. Dr. Johann Schlichter
	2. Univ.-Prof. Dr. Florian Matthes

Die Dissertation wurde am 21.01.2004 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 28.04.2004 angenommen.

All references to web resources (URLs) both in the main parts and in the bibliography have been verified at January 07, 2004, unless otherwise noted.

Abstract

Communication and collaboration based on the internet are important factors in business, research, and everyday life. The term virtualization denotes the phenomenon that more and more aspects of our lives take place online. In today's markets, companies have to be quick and flexible in order to be successful. One of the strategies to achieve this is the virtualization of organizations, leading to the abolishment of classical spatial and temporal constraints and to a greater flexibility. The dynamic collaboration of small, modular organizational units is the key idea of this strategy. The partnering problem becomes the pivotal point in such organization networks, raising the question of how to assess the trustworthiness of personally unknown potential partners.

Similarly, in online auction houses, customers often do not know whether to trust vendors with respect to the quality of the goods offered. Traditionally, such problems are solved by exploring the personal social network and looking for trusted persons who know the person or organization in question. Yet, due to the increasing variety of communication media, it is difficult to keep aware of all people in one's personal social network. Therefore it is necessary to support the management of social relationships.

The goal of this thesis is the development of a general framework for social relationship management. Starting from observations concerning the aforementioned virtualization tendencies, this work examines internet-based communication and shared information spaces with respect to the kinds of social network data that can be extracted from them. Existing approaches to social relationship management are discussed. Such systems, however, concentrate on only one or very few kinds of social relationships and thus only manage special aspects of a user's social network. Therefore, a general representation of social relationships is needed which allows for the combination of various kinds of relationships and sources of social network data.

On the basis of this analysis and the characterization of social relationships in terms of sociology, this work introduces a formal model of social relationships based on semantic web technologies. The main design goals of this formalization are fostering interoperability, independence from proprietary applications, extensibility, and integration of privacy protection.

Building upon the formalization of social relationships, a multiagent system for distributed relationship management is developed. Agents act on behalf of one or several persons and exchange relationship information in order to answer queries initiated by their users or by applications. Three query types can be distinguished:

- Exploring the social network up to a certain depth
- Checking if a relationship chain with certain characteristics from one person to another exists
- Retrieving relationship chains with certain characteristics from one user to another

With the help of these three query types both the problem of how to get a trusted estimation of another

person’s reputation and the problem of how to keep aware of all people in one’s personal social network can be solved.

The concepts developed in this work have been prototypically implemented and represent a comprehensive solution of the aforementioned problems of social relationship management in internet-based communication and shared information spaces.

Acknowledgements

This work emerged from my activities as a research assistant at the chair for Infomatics XI: Applied Informatics / Cooperative Systems (Prof. Schlichter) at Technische Universität München.

First and foremost, I would like to thank Prof. Dr. Johann Schlichter for giving me the opportunity for this work and for many helpful comments and suggestions, which have been substantial for the success of this work. I would also like to thank the second referee, Prof. Dr. Florian Matthes, for helpful comments.

This work could not have been done without the countless, fruitful discussions with my colleagues. I am grateful for an excellent and friendly working atmosphere and effective cooperation. Special thanks go to Georg Groh for bearing my countless disturbances and to Dr. Wolfgang Wörndl for viewing a preliminary version of this thesis.

I would also like to thank Helmut Schönenberger and all other staff members of UnternehmerTUM GmbH, for the excellent cooperation. Furthermore, I am indebted to Claudia Moranz and Michael Wagner for discussing psychological and economical aspects of this work.

Finally, I would like to thank Marion Bröer for proofreading and giving suggestions for improving my English.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Communication and collaboration based on the internet are important factors considering business, research, and everyday life. However, the personal social network of people, which constitutes an important aspect of social interaction still lacks exhaustive technical support. This chapter presents a quick overview of problems and perspectives of interpersonal relationship management and related support systems.

1.1 Background

Modern communication systems, first of all the internet and related applications, offer a variety of new possibilities for designing communication and cooperation. In the course of this ongoing development distributed forms of collaboration and organization increasingly gain importance: Classical questions about the optimal location for companies cannot be given a unique answer — some reasons for this are the intensification of resource dependencies of companies, changes of ecological or political general conditions and decreasing costs of telecooperation technologies that cause a re-evaluation of the four target dimensions costs, time, quality and flexibility (Picot et al., 1996, pp. 262). Spatial distribution of today's companies is as often a consequence of this process as of the extended access to human resources: the production of resources which can be transferred via telecommunication technologies can easily be spatially outsourced, as can be observed in the field of software development. Caused by the increasing re-focussing of companies on their core competencies we can observe a trend towards modularization (Picot et al., 1996, parts 5 and 6) which also often leads to spatial distribution of the modules. This culminates in the vision of the *virtual organization*, which will be discussed in more detail later in this work.

Also classical constraints of time are dissolved by the possibility of a company's spatial distribution. Building upon telemedia, global acting companies can offer their services round-the-clock or speed up their processes by exploiting different time zones. Besides, Krüger (1999) postulates an increasing shift of competition for temporal head-starts and recognizes speed as one basic precondition for market leadership.

Against this background, telemedia offer new organizational scopes for design and at the same time impose the need of developing usage strategies. A basic systematization is presented by the "Anytime/Anyplace matrix" (Fig. 1.1), which differentiates four types of situations according to the spatial and temporal distribution of the actors. Section 3.1 presents a more detailed discussion of the different CSCW (computer-supported cooperative work) systems.

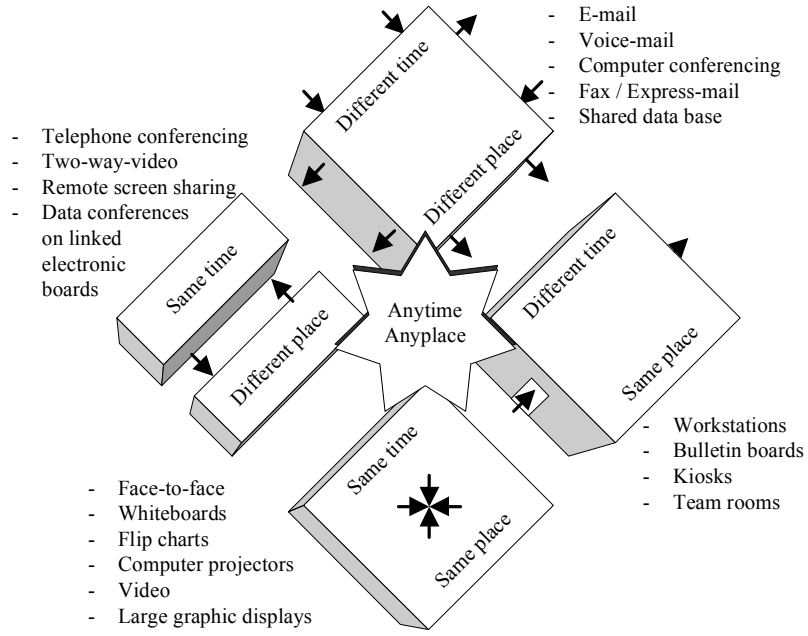


Figure 1.1: Anytime/Anyplace matrix (Reichwald et al. (2000, p. 7) according to O'Hara-Devereaux and Johansen (1994))

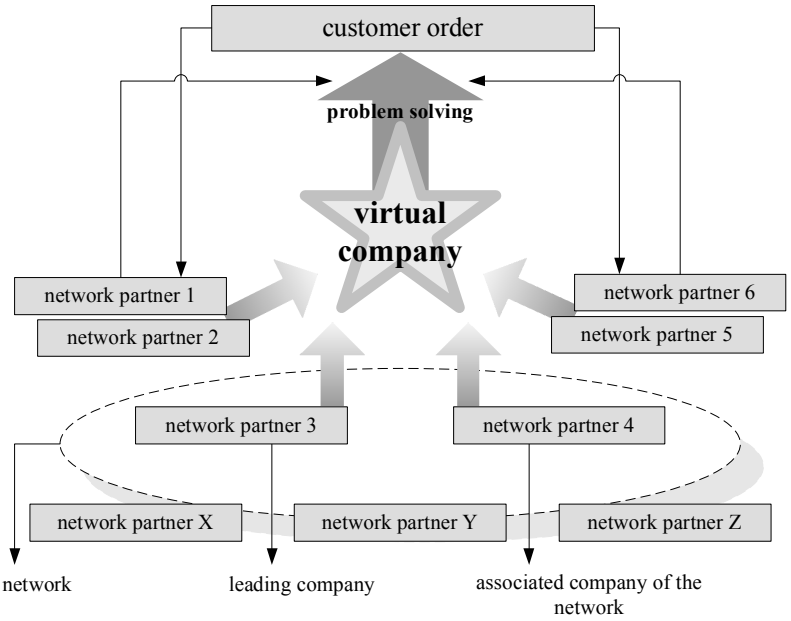
1.2 Virtualization

Building upon telemedia, many new concepts of organization and work structuring have been developed which heavily depend on the dissolution of spatial and temporal constraints. Some of these concepts are commonly subsumed under "virtual organization", though mainly two different complementary approaches can be distinguished (according to Reichwald et al., 2000, p. 254):

- According to the first approach, virtualization aims at the emergence of virtual realities, which allow for the replacement of concrete objects with electronic media, and makes it possible that places of employment, organizational structures, and companies overcome their physical constraints.
- The second approach in contrast focusses on virtualization as an organizational strategy which aims at increasing the dynamics of formal organizational structures and establishing a flexible network of workplaces, organizational units, and organizations.

If virtualization is particularly seen as a strategy to combine both approaches to form a new architectural paradigm for organizations, this proves to be a sustainable competitive strategy, as spatial and temporal constraints dissolve and the realization of complex coordination mechanisms is therefore less costly. This strategy offers an increase of capacities and flexibility. The following section presents a summary of this strategy, following the more detailed discussions of Reichwald and Möslein (1996, 1999).

Realization principles for virtual organizations. The starting point of every virtual organization is a concrete task whose solution requires the combination of individual resources from the network



A virtual company is initiated by a customer order. For solving the customer order, suitable companies associate to the virtual company. In some cases, a leading company is selected which takes responsibility for successful completion of the order.

Figure 1.2: Forming of a virtual organization, according to Wüthrich et al. (1997, p. 102)

forming the basis for virtual organizations. Usually the completion of each task only requires a part of the network. Once configured, each participant of such an organization contributes his own profile of skills and qualities to the solution of the initial problem. Referring to the whole network this may lead to an increase of capacities. Figure 1.2 illustrates the forming process of virtual organizations.

Virtual organizations are chiefly characterized by three realization principles concerning their main characteristics modularity, heterogeneity, and spatial and temporal distribution: According to the *open-closed-principle* a virtual organization appears at the market as a self-contained unit, in spite of its internal modular and open structure. The customer perceives a company which exactly matches his specific needs, although the forming process of the virtual organization is not apparent for the customer and does not take place until the problem solving process begins. The modular units of the so formed organization supplement complementary competencies. This is reflected in the *complementarity principle*. Finally, the concealment of the organization's modularity is mirrored in the *transparency principle*, as is the concealment of the localization and time flow of the problem solution.

Design objectives of virtual organizations. The aim that the virtualization of organizations pursues, is flexibility. The approach of virtual organizations to achieve flexibility differs from classical flexibilization strategies insofar as it questions the constancy of the whole organizational structure, whereas classical flexibilization strategies stick to the organizational structure and try to achieve improvements and better adaptability under constant general conditions. "Virtual size" in spite of "real smallness" (Reichwald and Möslein, 1999) is one of the main benefits which helps virtual organizations to gain flexibility. Because of the open-closed principle, a virtual organization possesses size when acting on the market and at the same time takes advantage of the flexibility of its small modular

subunits. But size is not only important when acting on the market. Size is also a precondition for investments in technologies and innovation, which become increasingly important under the general conditions of globalization of economic activities.

Owing to their heterogeneity, virtual organizations can achieve *generalism in their external activities*, though their internal modular subunits can benefit from high specialization. *Internal specialization* mainly offers cost benefits and advanced efficiency.

Limits of virtualization. Telemedia allow virtual organizations to go beyond many boundaries. Nevertheless this organizational form also has its limits. According to the principal-agent problem (e.g. Reichwald et al. (2000), p. 53 et sqq., or Picot et al. (1996)) the quality of service, efforts, and possibly hidden intentions of the partner are usually not verifiable in employer-contractor or employer-employee interactions when information is not uniformly distributed. Traditionally the resulting need for safeguarding is satisfied by formal contracts, whereas virtual organizations tend to avoid such contracts for time saving reasons and try to provide *trust* as a substitute for formal contracts. This makes trust the crucial coordination mechanism of virtual organizations and at the same time imposes substantial constraints on virtual organizations. The main problem now is how to establish long-term stable trust relationships in the dynamic and transient structures of virtual organizations (Koch et al., 2000). In other words, an *information problem* consisting in the difficulty of reliably assessing the partner's promises, credibility and reliability can be observed. Of course there are many other factors imposing boundaries for virtual organizations, e.g. legal insecurity or the problem of building a high-quality information base, which is a precondition for virtual organizations[1]. This thesis, however, will focus on relationship aspects.

1.3 Online Auction Houses and Information Systems

Apart from the organization strategy of virtualization another aspect of modern telemedia's influence on interpersonal interaction and communication shall be sketched in this section: transactions in online auction houses and the exchange of information in information systems based on the internet.

Online auction houses[2] provide a platform on the internet where their clients can buy and sell goods in an auction manner (Kollock, 1999). The platform provider just acts as a mediacy and does not himself accept any responsibility for correctly carrying out transactions between vendors and buyers — though of course there usually are terms of use which all users have to agree with before they may use the platform. Similar to the discussion in the preceding sections partners often encounter the problem of not knowing one another when initiating a transaction and therefore are faced with the risk that the partner might not comply with promises given before (e.g. concerning the quality of service). As spatial distances are often large between such partners, the truthfulness of these promises usually cannot be verified before the deal is accomplished.

A similar problem exists in open information systems, which offer their users the possibility to publish information which will then be accessible for other users. Examples of such systems are recommender systems and opinion platforms[3] (Koch et al., 2000; Resnick and Varian, 1997). The

[1] Especially the initiation phase of virtual organizations was subject of a research project of the munich technical university (Technische Universität München, project TiBiD). One main focus of research was the importance of trust in the initiation phase and the development of tools for supporting the initiation of long term relationships (cp. http://www.telekooperation.de/tibid/).

[2] e.g. http://www.alleauktionen.de, http://www.ebay.de, http://www.ricardo.de (closed for auctions since November 17, 2003), http://www.intoko.de

[3] e.g. http://www.dooyoo.de, http://ciao.de

reliability of the information published by other users in such systems can often not be verified objectively. Many online auction houses and opinion platforms therefore offer mechanisms for assessing transaction partners or information published by other users. Some of these mechanisms are subsumed under *reputation systems*, which will be subject of the following section.

1.4 Reputation Systems

The enormous amount of information in large information systems such as the internet (and systems building upon it) often leads to problems of how to choose relevant information for the respective task. A lack of time or simply the costs associated with an extensive search often makes it inefficient to scan all available information for their relevance with respect to the current problem. In some situations this might even be impossible as holds true for online auction systems (see above).

In order to make the choice of relevant and reliable information easier, many service providers like the online auction house eBay[4] deploy reputation systems. Such systems try to give an objective assessment of the reliability of information items or the author of those items (Kollock, 1999): at eBay registered users can offer items for sale by auction. Each user is identified by a pseudonym he may choose himself. After the completion of a transaction both users have the opportunity to rate the transaction partner with respect to his reliability. Ratings may be positive or negative. Positive ratings are added to an account associated with the user, whereas negative ratings are subtracted from this account. Other users can view the user's current account balance at all times, which is visualized by a star whose color reflects the total number of positive ratings given to the user. Each rating should also contain a natural language comment. All such comments for one user can be accessed on a detailed page which lists all comments and displays the number of positive ratings the respective user received from other users.

Although reputation systems are widely used, they entail a number of problems. First of all, persons whose reputation is mainly negative are seldom found (Koch et al., 2000), which may be due to the ease a user can get a new pseudonym for the respective service with. One possible solution is to display a pseudonym annotated with its age always. This problem is discussed more deeply by Friedman and Resnick (2000) and will not be subject of this thesis.

Another problem with reputation systems is how to avoid unfairly high ratings and unfairly low ratings: users might give one another or — via creating a second pseudonym for the same service — even themselves unfairly high ratings. Likewise it is possible that one person or a group of persons knowingly provide unfairly low ratings to other users in order to affect the reputations of those. These problems can partly be faced by hiding the mapping of pseudonyms to users and assigning random pseudonyms for each login, which makes providing unfair ratings more difficult. Besides, cluster algorithms may be used to filter out unfairly high ratings. Both approaches, however, have some limitations. For example, it may not be possible in all use-cases to hide a users true identity by assigning random pseudonyms. For a more detailed discussion of both approaches and their limits the reader is referred to Dellarocas (2000).

As a different method of resolving the problems mentioned above a reputation system might only consider ratings from users the user seeking information has positive experiences with when calculating the reputation of the user in question (Abdul-Rahman and Hailes, 1999). Lueg (1997) suggests that recommender systems should pay more attention to the social context and especially to the relationship between the authors of the ratings and the recipient of the ratings.

[4] http://www.ebay.de

Furthermore it is possible not to represent the rating itself but suggest possible information sources the seeking user has some relationship with (Granovetter, 1973). A support system implementing this approach focusses on collecting and structuring *relationship information*[5] between persons. Information sources can be email or transaction histories, but the analysis of publicly available newspaper articles or websites is also valuable with respect to inferring relationships between persons (Kautz et al., 1997a,b). The added value of a system like this consists in simply suggesting whom to ask in order to get the information needed. It is remarkable that indirect relationships — i.e. chains of direct relationships — can be integrated into this approach. Developing a general framework for *structuring and visualizing the personal social network* in communication systems and shared information spaces is the primary goal of this thesis.

1.5 Social Networks and Social Awareness

All approaches mentioned before have in common that they rely on interpersonal social networks. In the field of virtual organizations the social network is that between employees or decision-makers, or the social network between modular units or organizations themselves. The ties between persons or organizations consist in communication relationships, cooperation histories, personal or business acquaintances, formal or legal general conditions, interdependencies, etc. A similar list may be disposed concerning the social network between users of an online auction house, whereas the attention may be turned to relationships between persons concerning ratings and transactions, including their histories. These aspects have partly been given consideration in existing implementations, including the acquisition and the evaluation of transaction histories and interdependencies between vendors and buyers in online auction houses, the management of trusted key relationships in encryption systems like PGP[6], the extraction of information indicating interpersonal relationships from public documents in Referral Web (Kautz et al., 1997b), and the management of public or private buddy lists in virtual communities[7] and instant messaging systems[8].

The term relationship has been mentioned several times in the preceding sections without giving an exact definition. Chapter 2 deals with this term and related topics in more detail. The examples mentioned above may be seen as systems for management and evaluation of specifiic interpersonal or interorganizational relationships. It is important to put emphasis on the word “specific”: so far there are no approaches dealing with the information problems mentioned above in an exhaustive manner. Instead, those approaches focus on a specific application. This thesis tries to contribute to an exhaustive approach that takes into account interdependencies, interactions and communications in networks of persons in a more abstract manner, and is thus not restricted to one specific application area or support system.

Besides, it should not be overlooked that the analysis of social networks and their formalization and evaluation is also relevant in other application areas than that of seeking reliable information: sociometry, for example, deals with the analysis of relationships in groups of persons from the point of view of educational and industrial psychology. In applications supporting knowledge management an

[5] In this thesis, the term “relationship information” is used to denote data describing a relationship between two persons in an abstract manner. In contrast, the term “social network data” used by sociologists refers to concrete data obtained by measuring certain properties of network ties. Social network data can often be used for inferring relationship information.

[6] PGP uses a common hybrid encryption system where an asynchronous method is used for the exchange of a secret session key which is used for symmetric encryption of the message. Public keys can be signed by other persons in order to prove their trust in that public key.

[7] A more detailed discussion of virtual communities will be presented in Sect. 3.3.2.

[8] Instant messaging systems will be discussed in Sect. 3.2.4.

analysis of "Who knows who?" and "Who knows who knows who?" are of interest (Contractor et al., 1998). Sociology investigates the "small world problem", whose aim is the analysis of the length of relationship chains between a starting person and a target person[9] (Wasserman and Faust, 1994, p. 53). Nardi et al. (2002) report on a study suggesting that personal social networks increasingly gain importance for accomplishing work. At the organizational level, personal social networks are used for tasks like labor recruiting, partnering and information access. At the individual level social networks are relevant for finding new job opportunities and gathering information. In more detail, some of the tasks people reported in the study of Nardi et al. include:

- Remembering who was in their social network, particularly people who were important, but were contacted infrequently
- Remembering connections between different people in the network
- Remembering which documents had been exchanged with whom and when

For an individual user not only these social networks are important, but also his own embeddedness in and perception of them (Boyd, 2002, p. 60):

> Self-awareness allows users to understand who they are in a particular environment, how facets of their identity are manifested and aggregated, how other people and sites can see them. Such awareness places an individual within the society at large, in relation to other people.

Taking into account these problems, it is obvious that improved support tools for managing personal social networks are needed.

1.6 Goals and Roadmap of this Work

This thesis aims at contributing to an exhaustive approach for solving the information problems in social networks mentioned above. A formalization of interpersonal relationships is suggested, which can serve as a basic technology for interoperable relationship management in distributed communication and collaboration environments. Relationship information is information describing the personal social network of a person — i.e. that person's ties to other persons. Services and applications may collect relationship information during their use. For example, email-clients might extract relationship information from incoming and outgoing emails. The term interoperability suggests that relationship information can be exchanged between different systems. In particular relationship information collected by one service may also be used by other services. Apparently it is essential to provide means for transporting also the *semantics* of relationship information between different systems. This thesis suggests a solution inspired by semantic web technologies: The Resource Description Framework (RDF) and the Web Ontology Language (OWL) will play an important role in the representation of relationship information. In more detail, this work addresses the application scenarios described in the following two subsections.

1.6.1 Supporting the User in Assessing the Reputation of Other Users

In online auction houses or online market places, a user who is interested in buying an item from a vendor needs information about the vendor's reputation. In contrast to the classical approach of communicating the reputation of the vendor by collecting ratings from other users, this work suggests not

[9] The small world research project of the department of sociology of the Columbia University offers the opportunity of taking part in a small world experiment: http://smallworld.sociology.columbia.edu.

to transport an indicator of the reputation but to find people who can give an assessment of the vendor and with whom the buyer has a (possibly indirect) relationship. For that purpose, the relationship management system explores the social network of the buyer and tries to find a relationship chain to the vendor. The buyer should be able to define additional preferences like the kind of relationship to be included or the maximal length of the chain. Relationship chains that have been found by exploring social networks may either be directly used as a proof of trust in the vendor (e.g. if the relationship chain is a chain of trust relationships), or of contacting a trusted person who knows the vendor in order to ask for a rating of the vendor. A modification of the second approach is to assign weights to ratings of persons to whom a relationship chain of a certain kind can be found and to give higher weights to ratings originating from people with a close relationship to the buyer. This approach leads to a higher reliability of the cumulative rating of a vendor.

To summarize, the task a relationship management has to resolve in this scenario is: *Given a person x and a person y, find a relationship chain of maximum length n from x to y and take into account only relationships of a certain kind.*

1.6.2 Structuring the Personal Social Network

Tendencies of virtualization of traditional organizational structures lead to high dynamics of people's environments. Social networks become more and more important and replace traditional hierarchies. As reported by Nardi et al. (2002),

> people rely heavily on their own *personal social networks* to accomplish work. At the organizational level, personal social networks are activated for labor recruitment, partnering, and information access [...]. At the individual level, people exploit their networks to advance their own careers through finding new job opportunities and gathering information.

Recognizing the increasing importance of "networking", Nardi et al. (2002) identified "expensive" frequent tasks in networking. Some of these are the following.

- Remembering who is in one's social network — particularly people who are important but contacted infrequently
- Remembering relationships between people in the network
- Remembering which documents have been exchanged with whom and when
- Using multiple communication media easily

For this work, this boils down to the following tasks and requirements:

- *Starting from a person x, recursively explore the social network of that person up to a certain depth.*
- *In the preceding task, allow for the restriction to special kinds of relationship.*
- *Integrate relationship information of heterogeneous kinds that involves multiple communication media. Integrate relationship management with communication media.*

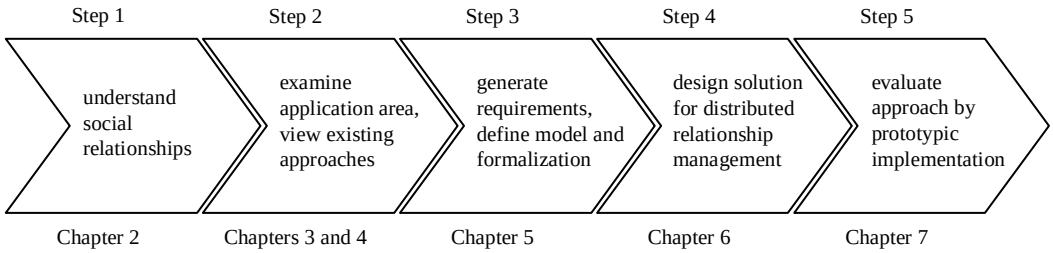


Figure 1.3: Roadmap of this thesis

1.6.3 Roadmap of this Work

On the basis of the two scenarios discussed above, a roadmap for this thesis can be set up, which is shown in Fig. 1.3. At first, it is necessary to understand the concept "social relationship". Questions herein are: how are social relationships characterized? What different kinds of social relationships are there? How can social relationships be measured?

Since this work addresses internet-based communication media and shared information spaces as its primary application area, as a second step social relationships in such systems must be examined. Which kinds of relationship do occur and where, how can specific kinds of relationship be measured, are questions addressed in this step.

On the basis of sociology and the application area, requirements for a formalization of relationships can now be defined. A model for social relationships must be developed on the basis of these requirements and implemented by choosing a suitable formalism. This is done in step 3.

Step 4 deals with the management of relationship information. Step 3 has defined *what* is managed, whereas in step 4, mechanisms for *how* it is managed must be developed.

Finally, an evaluation is done by prototypically implementing the core concepts of this work.

This work does *not* address the design of user interfaces. This aspect is, however, very important, because the technologies developed in this work turn out to be very complex. Future work must bridge the gap between high complexity and expressive power on the one hand, and a good usability and high security on the other hand.

1.7 Thesis Structure

This work is organized as follows (cf. Fig. 1.4): After presenting basic sociological notions for describing social relationships in Chapt. 2, Chapt. 3 gives a brief overview about computer-supported cooperative work and community support systems. Special attention is paid to the importance of managing relationships between users of such systems, and to the question which systems generate or use information about relationships. Internet-based communication and shared information spaces are the primary application area of this work.

Recently, the vision of the semantic web has become more and more popular in the world wide web community. Research on the semantic web aims at developing a rich framework which allows for higher expressiveness and improved automatic processing in the world wide web. Chapter 4 deals with technologies and visions of the semantic web, how these technologies can be applied to the management of relationships, and why it is useful to model relationships in the semantic web.

Based on the discussion of the application area, Chapt. 5 focuses on developing a model for in-

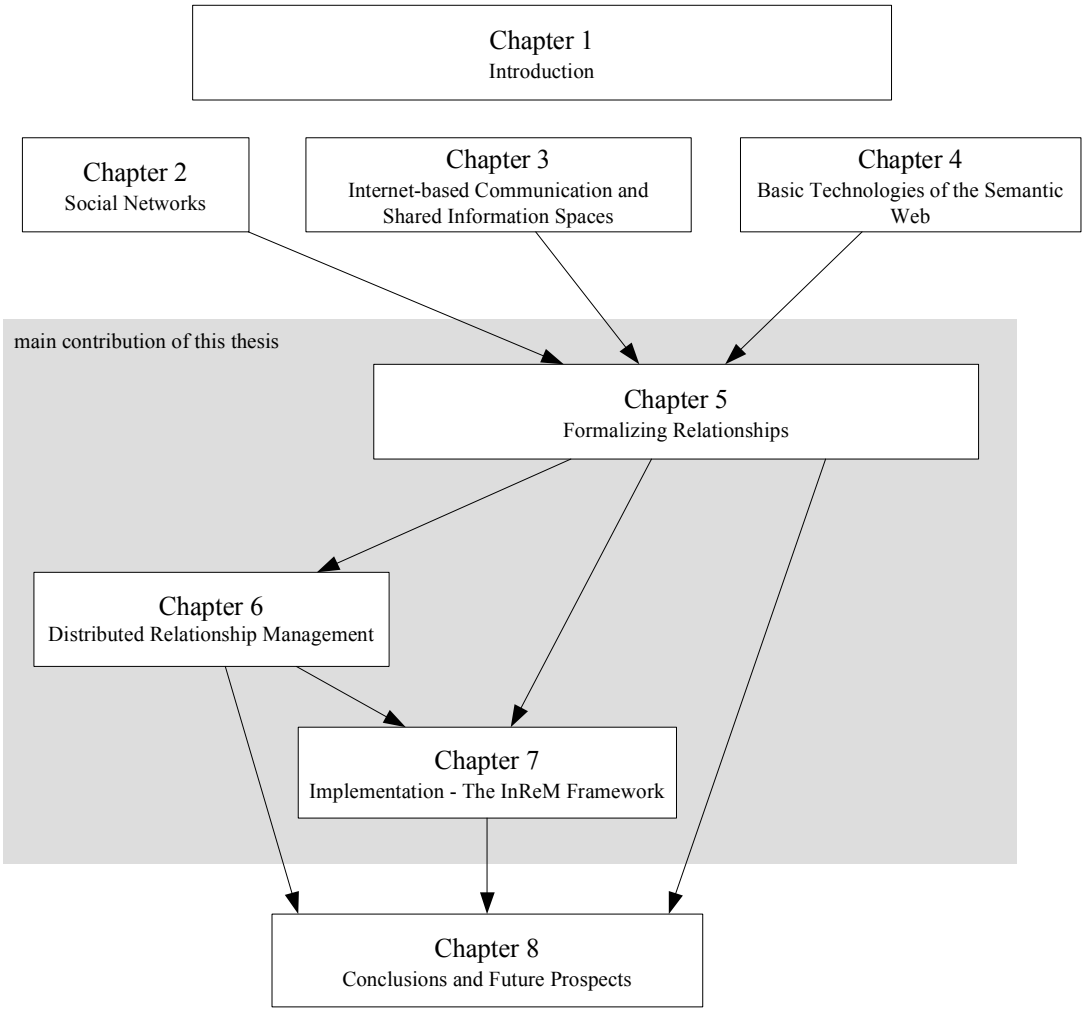


Figure 1.4: Thesis organization

terpersonal relationships. As mentioned before, technologies developed in the research area of the semantic web will be of use here. Ontology languages provide means for modeling domain theories which can be used for describing relationships. As an intermediary result, a relationship ontology vocabulary is defined offering the possibility of exchanging relationship information between different systems. Most notably, the relationship ontology vocabulary allows to express important characteristics of the relationship types.

Every person usually interacts with several systems. This even holds true when concentrating on one interaction partner: two people may use the email client of their own choice for sending emails to one another. At the same time these two people may use a shared group calendar for arranging meetings. As shown by this example, social network data concerning one person is usually distributed among several applications. Besides, there are services which need relationship information about groups of persons. The need to develop mechanisms allowing to draw conclusions in a distributed relationship information knowledge base is obvious, therefore Chapt. 6 deals with this problem. As a solution, the chapter suggests a multiagent system where each user possesses his own personal relationship information agent. The system also includes mechanisms for protecting personal relationship information against illegitimate access. The formalization of relationships itself can contribute to solving this problem. Traditional methods for privacy protection usually use artificial concepts like groups or access tickets, which have to be explicitly defined by system administrators. A privacy protection mechanism building upon relationship information may in contrast dispense with artificial concepts and grants access rights based on the relationship between the requesting person and the owner of the information.

Chapter 7 evaluates the concepts developed in Chapters 5 and 6 by presenting a concrete implementation of the core concepts discussed in the preceding chapters: the framework for interoperable relationship management InReM and its applications.

Finally, Chapt. 8 gives a summary of the most important conclusions of this thesis. The extensibility and transferability of the ideas is discussed. Interesting problems for future works are presented.

Chapter 2

Social Networks

Investigating social relationships and social networks is a main research focus in the field of sociology. This chapter introduces basic notions for describing social relationships and groups, which form the basis for the formalization of relationships presented in Chapt. 5.

2.1 The Sociological Perspective on Relationships

In everyday life the term "relationship" stands for a variety of different concepts. On the one hand, "relationship" denotes objectively measurable facts like legal or economic relationships. On the other hand, this term is used to describe subjective emotional ties between persons like friendship, liking, respect or trust. Sociometry focuses on the latter (Bjerstedt, 1956, cited by Dollase, 1979):

> Wir können die Soziometrie als ein Gebiet betrachten, das sich mit der Messung jedweder Art von zwischenmenschlicher und zwischentierischer Beziehungen befasst, wobei zur Zeit ein besonderes Schwergewicht auf die Erforschung zwischenmenschlicher Präferenzbeziehungen auf der Grundlage subjektiver Einschätzungen gelegt wird.

One problem in sociometry research is the development of measuring processes allowing for conclusions about social positions of the members of some group. Indicators are used which depend on the particular objective target of the sociometric test. Ties reflecting individual evaluation are put down to objectively measurable facts. Depending on the application area and the objectives of a sociometric test several methods for collecting data may be chosen. Two important examples are questionnaires and observations (Dollase, 1976). In the first case, questionnaires containing introspective questions are given to the members of a group. For example, questions might be "Which member of this group you would like to carry out your next project with?", or "Please rate each member of this group on a scale from 0 (worst) to 5 (best) according to her/his reliability." In observations, by contrast, no explicit interaction with the group takes place. This method focusses on systematically observing the group members' behavior in order to draw conclusions about relationships.

Sociometric tests are applied in various application areas. They are widely used for examining social structures in classes or work groups. This thesis will not discuss sociometric methods in more detail. For further information the reader is referred to Dollase (1976).

The main focus of sociometric research are emotional ties. Nevertheless, in this thesis these relationship types are only one aspect among others. In this chapter a more detailed review of basic notions is given from a sociological point of view, where relationships are classified according to the following types (Knoke, 1982; Wasserman and Faust, 1994):

- Individual evaluations
- Exchange of material goods
- Exchange of immaterial goods
- Interaction
- Formal roles
- Kinship

Individual evaluations have been mentioned before with regard to sociometry. It is possible to differentiate between positive and negative evaluations.

The second relationship type includes business transactions, imports or exports of goods, lending or borrowing, and contacts made in order to secure valuable resources.

Examples of relationships based on exchange of immaterial goods are communication relationships of any kind, for example, the sending or receiving of messages or passing on recommendations. It is possible to further differentiate between subtypes according to communication media (Garton et al., 1997).

Relationships based on interaction include all kinds of physical interaction of persons like attending the same event, sitting next to each other, visiting other persons' homes. All kinds of emotional support may be attributed to this type, too.

Formal roles reflect power or authority as can be observed between employee and superior, student and teacher, or patient and doctor.

Finally, the last type subsumes kinship relationships of various kinds. These include marriage and all kinds of family relationships as well as relationships between descendant and ancestor.

Some of the relationship types mentioned above cannot reasonably be represented in a computer-based support system. Emotional support, for example, usually cannot be appropriately captured in a computer system. Information concerning emotional support must be entered manually instead. Obviously, there are other relationship types which map into computer support systems more easily, for example, email communication, or interaction histories of a multi-user application. This thesis is restricted to those relationship types which can reasonably be represented in computer support systems.

Garton et al. (1997) give an overview of analyzing online social networks. Their attention is mainly focused on social networks supported by computer networks and computer mediated communication. The following paragraphs present basic definitions for sociological terms on the basis of Garton et al.'s discussion.

According to Garton et al. interpersonal relationships are characterized by *content*, *direction*, and *strength*. "Content" refers to the object being exchanged. Objects may be files, programs or more complex objects. With respect to communication, for example, objects may be meeting requests, emotional support, or other administrative or social matters. In electronic commerce, objects also may represent real world objects such as money, goods or services.

A relationship can be directed or undirected.[1] Sending a file from one person to another, for example, may be modeled in two different ways. First, a relationship exists from sender to recipient ("sending a message to"). A second relationship exists from recipient to sender ("receiving a message

[1] Directed resp. undirected relationships are later referred to as anti-/asymmetric resp. symmetric relationships, but in this paragraph, the notions of Garton et al. (1997) are adopted.

from"). In the case of undirected relationships the order of the arguments is irrelevant. For example, if one person works for the same company as another person, then the other person also works for the same company as the first person. Undirected relationships, however, may be unbalanced. One person may initiate communication more often than the other, or two persons' evaluations of the same relationship may be different.

Depending on the type of the relationship, the strength of a relationship reflects different aspects. Communication relationships might use the frequency of messages exchanged as a measure for strength. Other relationship types use the importance or complexity of information exchanged or the amount of capital transferred (Wellman, 1997; Wasserman and Faust, 1994). As has been mentioned in Chapt. 1, finding paths in social networks from one person to another can be important when looking for reliable information. Sociologists have proposed a variety of units of measurement for the strength values of paths of valued relationships (Yang and Knoke, 2001). The two most important approaches are the following:

Path value. This approach assumes that the value of a path between two persons is the lowest value of all path elements. If $p_1, ..., p_n$ is a path from person p_1 to person p_n, then this path is assigned the value $\text{pv}(p_1, ..., p_n) = \min_{i=1...n-1} s(p_i, p_{i+1})$, where $\text{s}(p, q)$ denotes the value of the relationship from p to q. This measure is occasionally referred to by "Peay's measure". Yang and Knoke (2001) refine this measure by dividing it by the path length n, in order to account for the costs required for involving a number of intermediaries:

$$\text{apv}(p_1, ..., p_n) = \frac{\min_{i=1...n-1} \text{s}(p_i, p_{i+1})}{n} \tag{2.1}$$

Path length. The path length approach, which sometimes is referred to as "Flament's measure" assumes that the values of relationships are costs. The value of the path is then defined as the sum of the values of all path elements: $\text{pl}(p_1, ..., p_n) = \sum_{i=1}^{n-1} \text{s}(p_i, p_{i+1})$. Again, Yang and Knoke refine this measure by dividing it by the length n of the path, which again reflects costs imposed by involving intermediaries:

$$\text{apl}(p_1, ..., p_n) = \frac{\sum_{i=1}^{n-1} \text{s}(p_i, p_{i+1})}{n} \tag{2.2}$$

It is interesting that according to the refined measures apv and apl optimal connections between two persons may involve paths longer than the minimal distance between the two persons. Besides, the two measures may suggest different optimal connections for two given persons, since they stress different aspects of valued relationships.

As defined by Garton et al. the term "tie" (between two persons) denotes the set of all relationships between two persons. It must be noted that this definition does not agree with that of Wasserman and Faust (Wasserman and Faust, 1994, p. 18 resp. p. 20), where ties are single aspects of a relationship. In this thesis the term tie is used according to the definition of Garton et al. As ties sum up relationships between persons, they also vary in content, direction and strength. Generally, ties are often referred to as "weak" or "strong", although there is no general definition for what "weak" or "strong" means. Nevertheless some important characteristics may be given. Weak ties are generally infrequently maintained and non-intimate. Strong ties include frequent contacts, close friendship and provision of reciprocal services. Yet, especially weak ties are important for extending the own social network, as is discussed in the following section.

2.2 Strong Ties and Weak Ties

In a remarkable paper, Granovetter (1973) discusses the strength of ties in interpersonal networks. Granovetter distinguishes between strong ties and weak ties, where strength is "a (probably linear) combination of the amount of time, the emotional intensity, the intimacy (mutual confiding) and the reciprocal services which characterize the tie". The main hypothesis of this paper is the following: if there exists a tie from a person a to a person c and also from a person b to c, then there most likely also exists a tie between a and b.[2] In other words, the configuration $(R(a,c) \wedge R(b,c) \wedge \neg R(a,b))$ is very unlikely to occur[3] (see below for an explanation of the notation). This hypothesis has important consequences for *bridges* in social networks: a bridge is a tie in a network, which provides the only link between two persons. If the network is large, then bridges are probably rare. Nevertheless, when restricting to subnetworks, by the same definition local bridges can be applied. Granovetter concludes that *bridges never are strong ties*. Consider the configuration $(R(a,b) \wedge R(b,c))$, where $R(b,c)$ is a bridge. If $R(b,c)$ was a strong tie, then there most probably would also exist a tie between a and c. Weak ties are essential links between subnetworks and represent connections to other social networks of (strong) ties. Thus weak ties provide means to facilitate moving from one social (sub-) network to another (Pickering and King, 1992).

Another important consequence of Granovetters hypothesis is that, since weak ties are "more effective in bridging social distance, [...] more people can be reached through weak ties" (Granovetter, 1973, p. 1369). Concerning the problems discussed in Chapt. 1 this suggests that weak ties are more likely to provide short paths to the person in question than strong ties, if a person cannot be reached via a direct tie. Thus, the strength of ties may be seen as an important variable for the application area of this thesis.

2.3 Algebraic Properties of Relationships

Relationships may possess basic algebraic properties (Hage and Harary, 1983; Wasserman and Faust, 1994). These properties are *reflexivity, symmetry, antisymmetry, asymmetry* and *transitivity*. In the following definitions well known notations from mathematics will be used: if R denotes a relationship, and a and b denote persons, then the fact that a relates to b is denoted by $R(a,b)$ (slightly informal, also the "$R(a,b)$ holds true" will occasionally be used in this thesis). The basic properties are defined as follows:

- A relationship R is reflexive, if $R(a,a)$ holds true for all persons a.[4]
- A relationship R is symmetric, if for all pairs (a,b) of persons $R(a,b)$ implies $R(b,a)$.
- A relationship R is antisymmetric, if for all pairs (a,b) of persons $R(a,b) \wedge R(b,a)$ implies $a = b$.
- A relationship R is asymmetric, if for all pairs (a,b) of persons with $a \neq b$ $R(a,b)$ implies that $R(b,a)$ does not hold true.
- A relationship R is transitive, if for all persons a, b, c, $R(a,b) \wedge R(b,c)$ implies $R(a,c)$.

[2] Granovetter presents some evidence for this hypothesis. A more detailed discussion, however, requires thorough definitions of the strength of ties and methods of analysis, which is beyond scope of this thesis.

[3] For this section, it is assumed that R is symmetric.

[4] Reflexivity is sometimes needed due to formal reasons, which will become clear in Sect. 5.1.

- A relationship R is non-symmetric, if it is neither symmetric nor antisymmetric nor asymmetric.

Note that every asymmetric relationship is also antisymmetric: because R is asymmetric, there is no pair of persons $(a, b), a \neq b$ with $R(a, b) \wedge R(b, a)$. Furthermore, for each person a, $R(a, a)$ implies $\neg R(a, a)$. Thus, there is no person a with $R(a, a)$. As has been mentioned before, asymmetric and antisymmetric relationships are sometimes also called directed relations. Likewise, symmetric are sometimes referred to as undirected relationships.

2.4 Groups

If one is interested not only in examining single relationships or ties between two persons but also between the members of a set of persons, the concept of a group can be helpful. Wellman (1997) defines a group as

> a social network whose ties are tightly-bounded within a delimited set and are densely-knit so that almost all network members are directly linked with each other.

Essentially, the following methods for modeling groups can be distinguished (Knoke, 1982; Wasserman and Faust, 1994):

- Groups based on undirected relationships
- Groups based on directed relationships
- Groups based on nodal degree
- Groups based on the strength of relationships

In the following, this thesis uses the notations of Wasserman and Faust, who employ basic notations from mathematical graph theory, where persons map to nodes of a graph and relationships map to edges between nodes. Any reader not familiar with these basic concepts is kindly referred to Wasserman and Faust (1994, Chapt. 4). However, section 5.1 will give exact definitions for those graph-theoretic terms needed for this thesis' modeling of interpersonal relationships. For a more exhaustive discussion of the different approaches for modeling groups the reader is referred to Wasserman and Faust (1994, Chapt. 7).

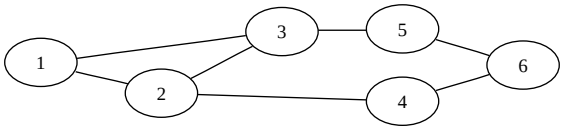
Cliques based on undirected relationships. A clique is a maximal[5] subset of the set of persons, where each person is related to every other person by a undirected relationship. Usually, only relationships of one special type are concerned. Every person may be a member of multiple cliques, so cliques may overlap.

The concept of a clique can be extended as follows: an n-clique is a maximal subset of the set of persons, where any two persons of the subset are connected by a path[6] (in the original graph) of the maximum length n.

Cliques as defined before are 1-cliques according to this definition. It is important to note that this definition causes n-cliques to have undesirable properties. Since in the preceding definition paths

[5] "Maximal" means that no additional persons can be added to the subset without violating the constraints constituting the subset (in this case: completeness of the subgraph).

[6] A path of length n from person a to person b is a sequence of persons $a, c_1, c_2, c_3, ..., c_{n-1}, b$, where a is related to c_1, c_1 is related to c_2, and so on, and c_{n-1} is related to b.



2-cliques: $\{1, 2, 3, 4, 5\}$ and $\{2, 3, 4, 5, 6\}$
2-clan: $\{2, 3, 4, 5, 6\}$
2-clubs: $\{1, 2, 3, 4\}$, $\{1, 2, 3, 5\}$ and $\{2, 3, 4, 5, 6\}$

Figure 2.1: n-cliques, n-clans and n-clubs (Wasserman and Faust, 1994)

between persons may also contain persons not included in the subset of persons constituting the n-clique, the minimal length of a path within the n-clique might be greater than n. Even worse, for some pairs of persons there might even exist no path within the n-clique.

In order to improve n-cliques, the following two additional definitions are given. An n-clan is an n-clique, where any two persons of the clique are connected by a path of maximum length n *within the n-clique*, whereas an n-club is a maximal subset of the set of persons, where any two persons of the subset are connected by a path of maximum length n within the subset.

What is the difference between these two definitions? n-clans are restrictions of n-cliques, so every n-clan also is an n-clique. A similar implication does not hold for n-clubs, because the condition that cliques are "maximal" subsets of persons refers to different constraints for n-cliques and n-clubs (see Fig. 2.1 for examples). However, it is true that every n-club either is an n-clique or a subgraph of an n-clique.

Cliques based on directed relationships. Concerning directed relationships, the definitions presented above have to be slightly modified. For directed relationships it is generally not true that for any relationship $R(a, b)$ between two persons a and b there also exists a relationship $R(b, a)$ with opposite direction. Of course the simplest solution to this problem is to explicitly demand for each relationship $R(a, b)$ the existence of a relationship $R(b, a)$: a clique based on a directed relationship of type R is a maximal subset of the set of persons, where for any two persons a, b in this subset both $R(a, b)$ and $R(b, a)$ hold. This, however, imposes heavy restrictions on cliques, leading to cliques being rare phenomenons when directed relationships are concerned. Hence it is useful to also give some weaker definitions for cliques based on directed relationships.

Two persons a and b are:

1. Weakly n-connected, if there exists a path between a and b of length n or less, where directions of relationships are ignored
2. Unilaterally n-connected, if there exists a path from a to b or a path from b to a of length n or less
3. Strongly n-connected, if there exists both a path from a to b and from b to a of length n or less
4. Recursively n-connected, if a and b are strongly n-connected, and the path from a to b uses exactly the same persons as the path from b to a, in reverse order

Based on these four types of connectivity four different kinds of n-cliques can be defined:

1. A weakly connected n-clique is a maximal subset of the set of persons in which any two nodes are weakly n-connected.

2. A unilaterally connected n-clique is a maximal subset of the set of persons in which any two persons are unilaterally n-connected.

3. A strongly connected n-clique is a maximal subset of the set of persons in which any two persons are strongly n-connected.

4. A recursively connected n-clique is a maximal subset of the set of persons in which any two persons are recursively n-connected.

Obviously, recursively connected 1-cliques are also cliques according to the preceding definition.

Groups based on nodal degree. Concerning undirected relationships, Wasserman and Faust present two methods for defining groups according to nodal degree:

- A k-plex is a maximal subset of the set of persons, in which each person is related to at least $g - k$ other persons in the subset (g denotes the cardinality of the subset).
- A k-core is a maximal subset of the set of persons, in which each person is related to at least k other persons in the subset.

It is clear that every 1-plex also is a 1-clique.

Groups based on the strength of relationships. When relationships possessing strength values are concerned, it may be useful to consider the strength for defining groups. The definitions presented above can easily be extended to relations with strength values by defining a mapping σ from the set of strength values to the set $\{0, 1\}$. Then a new relationship can be defined as follows. If $\chi_R(a, b)$ denotes the strength of the relationship R between a and b (if there is any), $R'(a, b)$ holds true if, and only if, $\sigma(\chi_R(a, b)) = 1$.

If strength values are real numbers and $c \in \mathbb{R}$, a common example of such a mapping is the following:

$$\sigma_c(x) = \begin{cases} 1 & \text{if } x \geq c \\ 0 & \text{otherwise.} \end{cases}$$

In this special case, an *n-clique (n-plex, n-core) at level c* referring to the original relationship R includes exactly those persons forming an n-clique (n-plex, n-core) with respect to the relationship R' obtained via σ_c.

2.5 Networks of Networks

Social network analysis is not restricted to single persons and relationships between persons, but can also be shifted to a more abstract level taking into account networks of persons (like groups) instead of persons themselves (Garton et al., 1997). In such "networks of networks", nodes may correspond to groups according to one of the definitions presented in the preceding section or to organizational units such as companies or work groups. Figure 2.2 clarifies the idea of shifting from single persons to groups.

How can relationships between networks be deduced from relationships between persons? A relationship between two networks may be assumed if there is a person a in the first network and another person b in the second network and there is a relationship from a to b. A special case is a

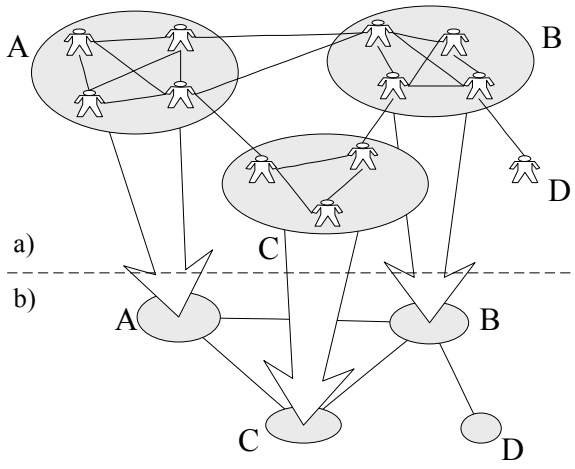


Figure 2.2: A network of networks, according to Garton et al. (1997). The upper part a) shows relationships between persons, the lower part b) shows relationships between the corresponding networks.

person being member of two or more networks — this also constitutes a relationship between those networks (Wellman, 1996; Garton et al., 1997; Wasserman and Faust, 1994).

Relationships between networks are not restricted to groups as defined in the preceding section. One might also be interested in relationships between networks corresponding to organizations or work groups, for example, when investigating the flow of information across organizational borders (Garton et al., 1997) or interdependencies between organizations.

2.6 Collecting Data about Relationships

How to collect data about relationships? Wasserman and Faust (1994, pp. 43–58) list the following methods for data collection about social networks:

- Questionnaires
- Interviews
- Observations
- Archival records

Questionnaires. Questionnaires are commonly used for collecting social network data. Sociologists differentiate between several types of questionnaires according to the type of questions and the possible answers. These details, however, are of no consequence for this work. In general, data collection via questionnaires includes explicitly questioning persons about their relationships to other persons. Thus user interfaces of buddy list systems in virtual communities or instant messaging systems like ICQ[7] may be seen as analogues of questionnaires (cf. Sects. 3.3.2 and 3.2.4).

[7] http://www.icq.com

Interviews. If social network data cannot be collected via questionnaires, interviews (either face-to-face or over the telephone) may be used instead. This method is irrelevant in the context of this thesis, since analogs of interviews in computer-mediated communication entail the need for natural language processing methods. Natural language processing, however, is out of the scope of this thesis.

Observations. Observing interaction dispenses with the need for directly addressing persons and may therefore be used when they are not accessible for questionnaires or interviews. Observations are, for instance, suitable for relationships between persons attending the same event. In the context of computer-supported cooperative work, observations coincide with archival records in most cases.

Archival records. Collecting social network data by analyzing archival records is especially important in the context of computer-mediated communication and computer-supported cooperative work, since in most such systems there exist lots of archival records about communication and interaction between persons or at least they can be created easily. An example is Referral Web (Kautz et al., 1997b,a), a research project using websites as sources for social network data. Websites are scanned for occurrences of names and based on that information, relationships between persons are inferred. Another example is the analysis of newsgroups and discussion forums (Smith and Fiore, 2001; Smith, 1999). Apart from name occurrences, the hierarchical structure of newsgroups and some discussion forums can also be used as a source for social network data. These and other examples will be discussed in more detail in Sects. 3.1–3.6.

Boyd (2002, p. 45) recognizes the

> immense amounts of data [people produce whenever they go online] about themselves without even realizing it (Behr 2002):
>
> External logs (web, IM): login time, duration, files accessed, referring website, connecting to what people
> Personal ISP logs: time/date/duration, tracked web contact access, email messages
> Profile data: age, sex, address, email, occupation, industry, income
> Affiliations: website/email domain
> Personal website content: links, interests, bio, photos
> External websites references to an individual
> Message content: email, chat, IM, SMS, Usenet/bboard posts, journals/blogs
> Sharable data: MP3s and other files
> Social networks: IM, email, chat, Usenet/bboard
> Presence data: IM buddy lists, Outlook calendars
> Shopping habits, browsing habits, recommendations
> Reputation as buyer, seller, advisor
> Archives of data over time: conversations, websites

Most of this data could be used for analyzing a person's social network. However, since much of this data is collected with the restriction of not being shared with other parties, the users' explicit consent to using such data for social network analysis is required.

2.7 Conclusions

Subsuming the preceding sections, the following conclusions about interpersonal relationships and their representation can be drawn:

1. There are different relationship types, which differ in meaning and attributes. For every relationship type, there may be subtypes restricted to specific aspects of the general type, for example, fixing the communication media for communication relationships.

2. Relationships may have basic algebraic properties: reflexivity, symmetry, antisymmetry, asymmetry, and transitivity.

3. For asymmetric or antisymmetric relationship types there may be corresponding inversions, e.g. sending and receiving of messages.

4. Relationships may possess additional attributes which characterize special aspects of a relationship, e.g. the topic of communication relationships.

5. Relationships may have a special attribute called “strength”. Evaluations of the strength of a relationship between two persons need not be equal for both persons.

6. There are various methods for describing groups in terms of interpersonal relationships. Vice versa, groups define relationships between their members, and persons being members of two groups constitute a relationship between those groups. *There is a duality of persons and groups.*

These statements will be referred to in Chapt. 5, where a formal model for interpersonal relationships in communication systems and shared information spaces is presented.

Chapter 3

Internet-based Communication and Shared Information Spaces

This chapter discusses the application area of this work — internet-based communication and shared information spaces. Existing approaches to relationship management and their limitations are discussed. The main focus is put on what kinds of relationship occur in the application area and how they can be measured.

3.1 Classification of Groupware Systems

The development of computer-based support systems for cooperative work opened an interesting field of research, which is characterized by a high level of multidisciplinarity. Computer-supported cooperative work (CSCW) includes aspects of computer science, organization theory, psychology, and sociology. One part of CSCW, which is concerned with software for supporting groups, is commonly known as groupware. Ellis et al. (1991) define groupware as:

> computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.

This thesis aims at supporting groups consisting of people which interact via electronic communication systems or shared information spaces. Such groups may have sharp borders (which may be imposed by organizational structures) as holds true for teams working together on a task, or may not have clear organizational borders as holds for the group of persons posting to a discussion board. For this thesis, a group consists of at least two persons exchanging information (on a personal basis) and thus influence each other (cp. Teufel et al., 1995). This definition implies that members of a group know each other. The reader should be aware that this definition is quite less restrictive than that imposed by the common usage of the term "groupware", which rather concerns *teams* working on a shared task. On the other hand, a *community* is a set of people who share something, but who do not necessarily know each other or interact on a personal basis (Schlichter et al., 1998). Communities are discussed in more detail in Sect. 3.3.2.

In this thesis the vast variety of groupware applications shall not be discussed in detail. However, groupware is strongly related with social networks insofar as many groupware applications collect data about their users' social networks and many groupware applications can be improved if a general framework for representing social networks becomes available. The following sections thus classify groupware applications according to Teufel et al. (1995) and discuss how the applications in the different classes can provide and use social network data in order to improve their services.

While reading the following sections, the reader should bear in mind a warning given by Grudin (1994): when designing groupware applications, special care to social taboos and existing social structures should be paid. Especially software collecting information about people's social networks risks violating social conventions. Hence, due diligence is required in order to protect everyone's privacy and not to misuse social network data. Privacy protection is a very important problem, therefore Sect. 6.5 shows how social network data can be protected against unauthorized use. Furthermore, as holds for all groupware applications, relationship management tools (if viewed as a class of groupware applications on its own) must be carefully integrated with existing social and political structures. Nevertheless, this thesis does not focus on these social and political aspects of integrating relationship management tools with existing structures but concentrates on the technical aspects of relationship management.

Teufel et al. (1995) classify groupware according to the three dimensions *communication*, *coordination*, and *collaboration*, and distinguish between four classes of groupware applications (see Fig. 3.1):

Communication. This class of groupware applications focusses mainly on supporting communication among group members that are spatially distributed. Applications in this class can further be divided in synchronous and asynchronous applications. Synchronous applications often offer audio and video transmission. A common example is Microsoft NetMeeting, which comes as a standard tool with the Microsoft Windows XP operating system. Also instant messaging systems like ICQ or MSN support near synchronous communication. The most common examples of asynchronous communication are email and bulletin boards.

Shared Information Spaces. Sharing information is often essential for successfully completing tasks. Shared information spaces support the exchange and evolution of information items. Mainly, two classes can be distinguished: hypertext systems and community support systems.

Workflow Management. Workflow management applications provide support for administrating and executing business process chains (workflows). In many cases, workflow management is used in connection with business process automation. Usually, a workflow contains a list of those people involved in the process and their roles with respect to the process. These roles may constitute relationships among the people involved in the workflow. Depending on the workflow model these relationships may be stated explicitly (e.g. in communication-oriented models, see below), or implicitly (e.g. in form-oriented models, see below).

Workgroup Computing. Workgroup computing subsumes those applications which support cooperation in groups of people sharing common goals, which are characterized by a rather low structural complexity. Applications in this class are typically highly specialized for the domain they have been designed for. Common examples are planning systems, decision and conference management systems, group editors, and distributed hypertext systems.

The following sections describe some groupware applications in more detail and give examples of systems generating and/or using social network data. Therein, special emphasis is put on the application classes communication and shared information spaces, since the prototypic implementations described in Chapt. 7 are built on selected applications in these classes. The two classes workgroup computing and workflow management are just sketched for completeness.

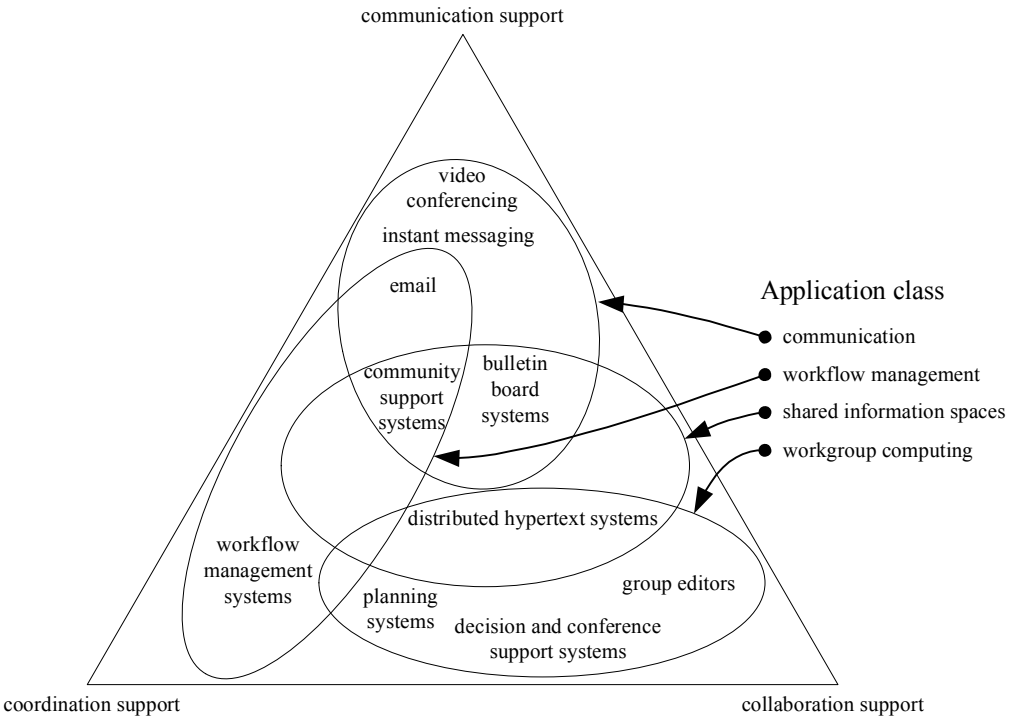


Figure 3.1: Classification schema for CSCW applications, according to Teufel et al. (1995).

3.2 Communication

In the application class communication, especially electronic mail, bulletin board, and instant messaging systems are interesting sources of social relationships.

3.2.1 Email

Electronic mail systems can be regarded as the most successful groupware applications and are widely used today. This success is due to several aspects as pointed out by Grudin (1994): email provides a benefit both for sender and recipient of messages (for the sender there is no extra work to do and the recipient can read the message at any convenient time). Email systems do not need a critical mass to be useful, even for two persons email can provide a benefit. Besides, the use of email allows for the application of social conventions. Email systems usually do not impose any structure on the communication process. Thus, email is used in a variety of contexts, ranging from formal job-related use to informal private use.

Wellman et al. (Garton et al., 1997; Wellman and Hampton, 1999; Wellman, 2001, 2002; Wellman et al., 2002) have shown that email offers a good insight into a person's social network. Most notably, the increasing use of the internet and internet-based applications like email does neither increase nor destroy social live but integrates with daily activities. Surveys show that a high percentage of people using email often are high in phone contact, too (Wellman et al., 2002). These observations document that email is a good indicator for the analysis of social networks. Email can furthermore help to extend and link social networks due to the ease with which messages can be forwarded to other persons. The forwarding of messages allows indirect ties to become direct ties and thus fosters the integration of

social groups (Wellman, 1996).

Recognizing email as a good starting point the analysis of social networks based on communication tools, Nardi et al. (2002) developed ContactMap. The primary design objective of ContactMap is to integrate both information and communication capabilities in a single user interface. Contact map analyzes the user's history of email interaction in order to gain social network data. In contrast to traditional social network analysis, ContactMap does not use this information for the generation of network graphs, but assists the user in arranging his contacts (which have been extracted from the email analysis) according to his own perception of his social network. For each contact found, the user may decide whether to include him or not. Additionally, ContactMap supports the association of contacts to groups. Currently, ContactMap does not support the exchange of data between two instances. The information gathered by two ContactMap installations of two different users therefore cannot be shared and ContactMap cannot be used to retrieve information about contacts of contacts. Another drawback is that currently only email relationships are considered. However, Nardi et al. have recognized these limitations and scheduled them as future work.

There have been several other approaches of analyzing email traffic in order to draw conclusions about social networks. Eick and Wills (1993) modified the standard UNIX mail program so that it logs each message it deals with. Based on this data, they ran network analysis and spring system visualization algorithms[1]. Eick and Wills observed that according to link strengths (which resemble the number of messages exchanged between two persons) groups with frequent email traffic could be identified. Another conclusion based on observing email traffic for several months is "that it takes about two months for most [newcomers] to establish communication patterns and settle in to their new position" (Eick and Wills, 1993).

The goal of PostHistory (Viegas, 2002) is to provide an opportunity for reflection about and insight in personal email usage patterns by visualizing one's personal email history. PostHistory only uses email headers (from, to, cc, bcc), not the message body itself. These message headers are then interpreted with respect to who knows who, contact strength and responsibility of a user in relation to other people in his network. The main perspective of PostHistory is to perceive how email contacts develop over time — especially when new contacts appear and when ties fade out. Like ContactMap, PostHistory is a personal tool and does not exchange data with other instances of PostHistory. Hence, though PostHistory assists with managing one's personal contacts it does not assist with managing one's personal social *network*.

Visual Who (Donath, 1995) aims at visualizing communication patterns based on mailing lists instead of individual emails. Visual Who runs on UNIX systems and takes standard UNIX alias files as input. If there are a total of n lists, for each list i, a profile vector p_i is calculated which has n entries, one for each list:

$$p_i^j = \frac{(N(i \cap j))^2}{N(i)N(j)}, 1 \leq i, j \leq n,$$

where $N(i)$ is the number of people subscribing to the list i and $N(i \cap j)$ is the number of people subscribing to both lists i and j. Profile vectors can be viewed as subscription-based similarity measures for mailing lists. For each person k and each list i, a resemblance is then calculated, which is the sum of the profile vectors of all lists the user is a member of:

$$R(k, i) = \sum_{j=0}^{n} M(k, j) p_i^j,$$

[1] Spring system visualization algorithms can be used to draw graphs of nodes connected with valued ties ("springs"). The values of the ties are used to calculate a spring constant for each tie and the nodes are arranged such that an equilibrium is reached (Eick and Wills, 1993).

where $M(k, i)$ is the membership function that returns 1, if person k is on list i and 0 otherwise. The main idea of Visual Who's visualization is the simulation of a system of springs, where persons are attached to mailing lists by springs. For a spring tying person k to mailing list i, $R(k, i)$ is taken as spring constant. It is important to note that the user running the visualization tool can choose, which mailing lists are to be displayed and that springs are set up only for those lists. Once started, the system simulates the movement of persons between mailing lists step by step, trying to reach equilibrium. Because a constant of friction is used, the movement will eventually halt. For screenshots of the visualization, the reader is referred to Donath (1995).

There are some points that are noteworthy with respect to this thesis. First of all, this example illustrates that mailing lists can provide useful information for social network analysis. But even more interesting, Visual Who shows that analyzing simple mailing list structures can yield interesting information about group membership: Donath reports on a situation where the visualization algorithm placed a professor right in a cluster of students associated with a mailing list about skateboarding. The professor, however, probably had never posted to the skateboarding list but it turned out that most of the skateboarders were musicians and the professor was a professor of music, who was involved in many projects together with the skateboarding musicians. Therefore, the visualization was quite right. In this way, Visual Who helps to understand the structure of the social network and points out relationships to other persons and groups one might not be aware of.

Social Network Fragments (Boyd, 2002) is another approach to visualize email communication. The data input consists of the message history and some additional information about which email addresses are actually mailing lists and a collapsing of all email addresses associated with one individual. The main difference to the approaches described above is that Boyd deals with different headers of the message in a more subtle way. The headers analyzed are to, from, cc and bcc. Based on this information, five categories of ties are distinguished:

Knowledge ties. If a person a sends a message to person b, then it is assumed that a "knows" b. (If b receives the message via a mailing list, then this is not assumed.)

Awareness ties. If b receives a message from a, it is assumed that b "is aware of" a.

Weak awareness ties. If both b and c receive a message of a via the to or cc header, it is assumed that b and c "are weakly aware of each other".

List awareness ties. If b receives a message from a via a mailing list, it is assumed that b "is mailing list aware of" a.

Trusted ties. If a sends a message to b and blind carbon copies (bcc) d, it is assumed that a "knows" and "trusts" d, because d has the ability to respond and reveal that a included people without the other recipients' awareness.

Each type of tie is given an importance, such that trusted ties have higher values than knowledge ties, which have higher values than awareness ties, etc. Values are additive, such that more intense conversation yields higher values. Based on this social network data, Social Network Fragments draws a graphical visualization which is based on simulating a system of springs, where springs connect persons and spring constants are the added values of ties between persons. By adding a default repulsion between all persons the system is prevented from collapsing to one point.

Like PostHistory, Social Network Fragments primarily aims at visualizing communication networks over time. Therefore, the Social Network Fragments visualization tool provides an animation

Email header	Explicit ties	Implicit ties
From: A		
To: X,Y	A ⇒ X	X ⇒ Y
	A ⇒ Y	Y ⇒ X
CC: Z	A ⇒ Z	X ⇒ Z
		Y ⇒ Z
		Z ⇒ X
		Z ⇒ Y

Table 3.1: Ties derived from an email header, according to Ogata et al. (2001)

of the spring system where one second corresponds to one day, such that people can watch their contacts appear, settle and move in their social network.

Ogata et al. (2001) use email traffic in order to analyze social communication networks. This social network data is used by a prototype (called Peco-Meditaor-II) for mediating cooperation — especially providing assistance with solving problems — in a work group. Ogata et al. (2001) extract explicit and implicit relationships from email-headers (see Tab. 3.1). Each explicit tie is assigned a strength value according to the formula

$$M_{i,j} = \left(\sum_{k=1}^{P} w_k \cdot \frac{n_{ij}(k)}{S_i(k)} + w_k \cdot \frac{n_{ji}(k)}{R_i(k)} \right),$$

where

$$w_k = \frac{P - k + 1}{P(P+1)}$$

is the weight of period k if P past periods are accounted for, $n_{ij}(k)$ denotes the number of messages sent from person i to person j in period k, and $S_i(k)$ and $R_i(k)$ are the number of messages sent by i in period k and received by i in period k, respectively. The main idea of this formula is that a tie is strong, if emails are exchanged frequently, recently, and reciprocally. The version of the formula presented above is slightly different from the original formula of Ogata et al. (2001). Two constant factors have been left out so that the formula yields values between 0 and 1. $M_{ij} = 1$ indicates that all of i's emails were sent to j, and i received emails from j, only.

Note that M is not symmetric in i and j. If a symmetric measure is required, the arithmetic average of $M_{i,j}$ and $M_{j,i}$ can be taken:

$$s_{i,j} = \frac{M_{i,j} + M_{j,i}}{2} \tag{3.1}$$

All systems described above have in common that they take a rather statistical approach which aims at visualizing email communication networks. These systems, however, do not model the concept of a relationship between two persons as such. Although this approach is certainly useful for the visualization, the obvious disadvantage is that it does not allow for the exchange of social network data between different instances of an application with reasonable effort. Therefore, the analysis of "who knows who" can only be done in a centralized manner, where all social network data is gathered in one place and analyzed with statistical means. Yet, a centralized approach has serious drawbacks: people cannot directly control what is done with the data they provided — email logs are certainly private information and people are usually not willing to make them publicly accessible. Additionally,

the knowledge generated by analyzing email logs cannot be shared with other applications (like email clients or personal electronic address books) as long as there does not exist a "universal" modeling of email communication relationships.

3.2.2 Usenet

Usenet news is a distributed decentralized bulletin board system whose origin dates back to 1979 (Pfaffenberger, 2003; Rheingold, 1993). First versions of Usenet were implemented as UNIX scripts until in the mid-1980s the Network News Transport Protocol (NNTP) was developed, which provided faster exchange of messages and improved control for server administrators (Kantor and Lapsley, 1986). Usenet servers store local copies of Usenet messages, which can be read by users via various Usenet clients, such as Microsoft Outlook or Mozilla Mail. Besides reading messages, users can post messages and replies which are then propagated to other servers. Messages are hierarchically organized in newsgroups reflecting different topics. Some examples of newsgroups are `comp.lang.c++` which contains messages about C++ programming, `rec.travel.asia` for posting messages about traveling Asia or the vast hierarchy of `alt.*`-newsgroups containing newsgroups for almost every special topic. Besides, there are localized newsgroups as those starting with `de.` which contain German messages. The number of Usenet newsgroups is growing. As reported by Miller et al. (2003), in 2001 there were more than 60,000 newsgroups and about 1.3 million messages per day. Estimates for the number of newsgroups vary, however, since this number is hard to determine due to Usenet's decentralized organization. Google tries to provide an entire archive of Usenet discussion groups dating back to 1981.[2]

Usenet is decentralized: there is no central authority managing the exchange of messages. Instead, each Usenet server decides itself which messages it stores locally and which ones it propagates to other servers. Thus, two users' views on Usenet may be different depending on the servers their clients connect to. When a message is posted to a newsgroup it is passed from server to server without any central control. Hence, the path a message takes from its author to a reader cannot be determined in advance but depends on the local propagation strategies of the servers on the message's way and the servers the author and the reader choose to connect to (Fisher and Lueg, 2003).

A newsgroup message m_1 may contain a reference to another messages m_2 indicating that m_2 is a reply to m_1. Similar to email messages, each newsgroup message header contains a (unique) id, which is used as a reference in replies. Fig. 3.2 depicts an example of a Usenet message which is a reply to some other message. Usenet message headers are defined in RFC 1036 (Horton and Adams, 1987). RFC 2076 (Palme, 1997) and RFC 822 (Crocker, 1982) contain additional information about possible message headers. The following headers of Usenet messages are especially important with respect to relationship management.

From. The `From`-header identifies the author of the message. The content of this header must conform to the email address format defined in RFC 822.

Message-ID. This header is used to uniquely identify a message. Message-ID's may not be reused in the lifetime of a previous message with the same Message-ID. According to RFC 822, the `Message-ID`-header must have the format `<local_id@full_domain_name>`, where `local_id` is a locally unique identifier (e.g. a number) and `full_domain_name` is the name of the host the message was posted to.

[2] http://groups.google.com/

References. The presence of this header indicates that the message is a reply to another message, which is identified by the content of this header. If the original message contains a reference header, the reference header must contain the Message-ID of the original message, a blank and the previous reference. Although RFC 1036 allows shortening the `References` header, this is not encouraged in practice.

In-Reply-To. For Usenet messages, meaning and syntax of this header are the same as for `References`, i.e. both headers indicate that a message is a reply to the message(s) listed. Although RFC 1036 defines `References` for this purpose, the `In-Reply-To` should be used preferably, as RFC 1036 is intended to be compliant with RFC 822, which only allows `In-Reply-To` in this case.[3]

A considerable amount of research has been done on the social impact of Usenet message exchange. Starting with Netscan from Microsoft Research, some of these approaches will be mirrored in the following paragraphs.

Netscan is an experimental tool for analyzing Usenet communication developed at Microsoft Research (Smith, 2003; Smith and Fiore, 2001; Smith, 1999). Since 1996 the system has been collecting data about Usenet usage, which can be accessed via a web interface.[4] The goals of the Netscan project are both providing empirical data about Usenet and developing enhanced interfaces to Usenet which take into account the social context of messages. "What's going on and where and who is around?" are some of the questions not sufficiently answered by common newsreaders. It is not easy for people to identify newsgroups of interest and to assess the history of participants or estimate how many people are active participants in a newsgroup. Also patterns of interaction may be of interest: does a newsgroup contain vivid discussion among participants or are there few "experts" replying to many messages but not initiating threads themselves? Without this information, both finding an appropriate newsgroup and establishing "trusted" relationships to other Usenet participants may be hard (Smith, 2003). The Netscan project aims at developing improved interfaces for Usenet addressing these problems. Reports and interfaces available in the public prototype include:

Newsgroup Grid. This report is the Netscan main page and offers an overview on a set of newsgroups whose name match a query which can be entered at the top of the page. The data presented includes the number of postings, posters, replies, and repliers in each group. The report period may be chosen between day, week, or month.

Newsgroup "Report Cards". For every newsgroup Netscan provides a "Report Card" which offers more detailed information about the group, such as a timechart for every day of the selected period, average message length or number of posters and a list of related groups, where relations between newsgroups are determined by analyzing cross-posting patterns (see below).

Thread and Message Browsing Interfaces. The thread and message browsing interfaces enable navigating threads and messages in a graphical tree representation. Every reply is connected by a line to the original message. By clicking on a message symbol, all other messages in the current thread posted by the same author are highlighted.

Author Profile. The author profile report provides information about which newsgroup an author has posted to and how often, how many threads he has touched and how many days the author has been active.

[3] RFC 822 also defines `References`, but the intended meaning is that of `See-Also` in RFC 1036. There is an Internet Draft known as "son-of-1036" (Spencer, 1994) which deals with these issues.

[4] The experimental Netscan prototype can be found at http://netscan.research.microsoft.com/.

```
Path: informatik.tu-muenchen.de!not-for-mail
From: Michael <mic267g@hotmail.com>
Newsgroups: de.rec.outdoors
Subject: Re: Wandern in Norwegen oder Schweden?
Date: Fri, 02 May 2003 10:56:25 +0200
Organization: Technische Universitaet Muenchen, Germany
Lines: 21
Message-ID: <b8tbut$5ffla$1@sunsystem5.informatik.tu-muenchen.de>
References: <20030428204846.4a63ee65.nobbie@web.de>
NNTP-Posting-Host: atschlichter56.informatik.tu-muenchen.de
Mime-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 8bit
X-Trace: sunsystem5.informatik.tu-muenchen.de 1051865885 5750442
    131.159.24.64 (2 May 2003 08:58:05 GMT)
X-Complaints-To: usenet@in.tum.de
NNTP-Posting-Date: Fri, 2 May 2003 08:58:05 +0000 (UTC)
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.3)
    Gecko/20030312
X-Accept-Language: en-us, en
In-Reply-To: <20030428204846.4a63ee65.nobbie@web.de>
Xref: informatik.tu-muenchen.de de.rec.outdoors:51018

Servus,

Ich würde mich der Empfehlung für Jotunheimen anschließen. Das ist zwar
die alpinste Region und damit gewissermaßen auch eine der
anstrengendsten in Norwegen, aber ich hätte keine Bedenken das
"Einsteigern" zu empfehlen. Ist halt so am interessantesten, und man
kann auch gleich die beiden höchsten Berge mitnehmen, Glittertind und
Galdhoeppigen.

Hardangervidda fand ich persönlich etwas langweilig, ist halt groß und
"flach". Es kann einem passieren, dass man den ganzen Nachmittag lang
auf die hütte zuläuft, und sie nicht so recht näherkommen will ;-)

Abisko in Schweden ist auch ein guter Ausgangspunkt, insbesondere kann
man das mit der sehenswerten Bahnfahrt nach Narvik verbinden und von
dort aus die Lofoten besuchen, die sich zum Tourenwandern aber nicht
eignen, wohl aber für Tagesausflüge (wobei ein Fahrzeug von Vorteil
wäre).

Mit dem Scanrail-Ticket bekommt man dort, wo keine Züge fahren, auch
Ermäßigung auf die Überland-Busse.
```

Figure 3.2: A Usenet message posted to the newsgroup `de.rec.outdoors` which is a reply to the message identified by the headers `In-Reply-To` and `References`.

Cross-posting patterns. Cross-posting analysis takes into account that message threads need not be restricted to one newsgroup. During its lifetime a thread may touch various newsgroups as its focus shifts. Netscan analyzes these cross-posting patterns and calculates cross-posting-relationships between newsgroups.

The report on interpersonal connections described by Smith and Fiore (2001) is missing in the current prototype. Nevertheless, it is very interesting with respect to this thesis as it visualizes interpersonal connections between authors in a thread. In that report, authors are arranged in a two-dimensional space where the x-axis indicates the number of replies posted by each author and the y-axis reflects the number of messages that are posted in reply to their messages. If one author replied to some other author, these two persons are connected with a line. The thickness of the line is determined by the number of replies exchanged between these persons.

There are some other reports and visualizations like the TreeMap visualization of Usenet providing interesting information for a sociological analysis of Usenet. With respect to this thesis, however, those reports and visualizations are of little importance.

Similar to Netscan, the goal of Communication Map (Sack, 2003) is to provide an alternative to traditional Usenet browsers including the social context of messages.[5] The graphical user interface includes a social network pane, which displays connections between participants in the selected group. Persons are displayed as nodes with optional labels. Two persons are connected by a line, if they have reciprocally replied to each other. The more frequently two users have replied to each other, the closer the corresponding nodes are arranged. The interface supports interactivity, such as clicking on nodes and thus highlighting portions of the social network.[6] Another interesting aspect of Sack's work is the "semantic network" display of terms used in message bodies. Sack employs text analysis procedures including linguistic databases like WordNet (Miller, 1995) in order to identify important terms in every message. These terms can then be used for a topic-based navigation through the set of messages in a newsgroup.

The Loom2 project (Donath et al., 1999; Donath, 2002; Donath et al., 2001) aims at developing visualizations of Usenet newsgroups in consideration of social structures of groups.[7] More specifically, some of the questions the Loom2 project tries to address are "How many members post in a given group?", "Do a few individuals dominate the group?", "How active is a group?", or "How many groups are people involved in?" (Donath et al., 2001). Although these questions concern the social structure of Usenet, Loom2 does not address issues of social networks directly. Nevertheless, graphical Loom2 spring visualizations (Donath, 2002) contain implicit information about the social network between Usenet participants, for example, the gathering of people to circles representing conversations. Size and density of circles correlate with the vibrancy of conversation threads. A newcomer can easily find vivid threads in a group and thus more easily identify important actors in the social network of that group.

3.2.3 Visualizing Usenet Relationships

In order to get a better impression of what social networks based on newsgroup relationships look like I did some empiric research. It must be emphasized that the analysis reported on in this paragraph is far from being representative of social networks in newsgroups in general. A more representative analysis

[5] Communication Map was designed and implemented by Warren Sack at UC Berkeley. A prototype can be found at http://www.sims.berkeley.edu/∼sack/cm/.

[6] A similar approach is reported on by Fisher (2000).

[7] The Loom2 project page is situated at http://smg.media.mit.edu/projects/loom2/.

requires a thorough choice of samples and more exhaustive analysis methods, which is beyond the scope of this thesis. Nevertheless, I think the results presented below provide a first insight in social networks in newsgroups.

The sample the following analysis is based on was collected from May 16 to May 22, 2003. It contains those messages the news server news.lrz-muenchen.de provided via anonymous access from inside the TU network[8] for the newsgroups `de.rec.outdoors`, `de.rec.alpinismus`, and all groups in the `tum` hierarchy (`tum.*`-groups are dedicated to discussing issues concerning Technische Universität München). Based on this sample, relationships are evaluated as follows. A message x is a reply to a message y if either x contains a header line `In-Reply-To` whose latest element is the Message-ID of y, or if this is not the case, x contains a header line `References` whose latest element is the Message-ID of y. Persons are identified by their email-address, where different email-addresses are assumed to refer to different persons.[9] The relationship *replies-to* between two persons a, b can be defined in two different ways. First, there is a directed relationship $R_d(a, b)$ from person a to person b, if a is author of message x, b is author if message y and x is a reply to y. $R_d(a, b)$ has strength n, if n such pairs (x, y) of messages exist. Second, there is a reciprocated relationship $R_r(a, b)$ between a and b, if there exist two sets of messages X and Y, where X contains all those messages of a which are replies to any messages of b (not necessarily in Y), Y contains all those messages of b which are replies to any messages of a (not necessarily in X), and both sets are not empty. $R_r(a, b)$ has strength m, if $m = \text{card}(X) + \text{card}(Y)$. Figures 3.3 and 3.4 show graphs G_r of R_r and G_d of R_d respectively, based on a data sample ranging from May 16 to May 22, 2003 for the newsgroup `de.rec.outdoors`.[10]

The first impression of both graphs is that they are pretty densely connected. The undirected graph G_r contains six cohesive subgroups where the biggest contains 26 nodes, two contain three nodes and the three small ones each two nodes. Strength values of most edges are rather balanced, i.e. the numbers of messages sent in each direction are about the same — in most cases one message in each direction. An exception is dyad $(4, 30)$, where person 30 posted three messages vs. one message of person 4. The most active persons (2, 5, 8 and 18) can easily be identified. Most remarkably, person 2 is central to this group of persons and has rather strong relationships to persons 5, 8 and 18 (of strength 1:1, 2:3, and 5:4 respectively).

The graph G_d is much bigger than G_r, however, many persons have just received or sent one single message. Person 2 is also central in this graph insofar as it has received several additional replies (from 1, 13, 24, 32, 37, 42, and 93) and has sent additional messages (to 12, 14, 20, 22, 62, 70, and 107) and thus is socially the most active person in this newsgroup. On the other hand, three persons (62, 70, and 87) can be identified who mainly receive replies. About two-thirds of all persons have only posted one reply message, and about half the persons have received only one reply. Most non-reciprocated relationships have strength one (in Fig. 3.4, for the sake of lucidity, strength values are not indicated). However, it is questionable if such edges should be considered as relationships at all — sending a single reply to someone will probably not lead to remembering that person for a long time. Hence, when interested in answering questions like "who knows whom?", reciprocated relationships probably provide better clues.

[8] The sample of newsgroups offered by the server depends on the location the client connects from.

[9] Sect. 6.2 will deal with the problem of having multiple identifiers per person.

[10] The layout of both figures was done with the Graphviz package from AT&T Labs Research (Gansner and North, 2000). Further information is available at http://www.research.att.com/sw/tools/graphviz/.

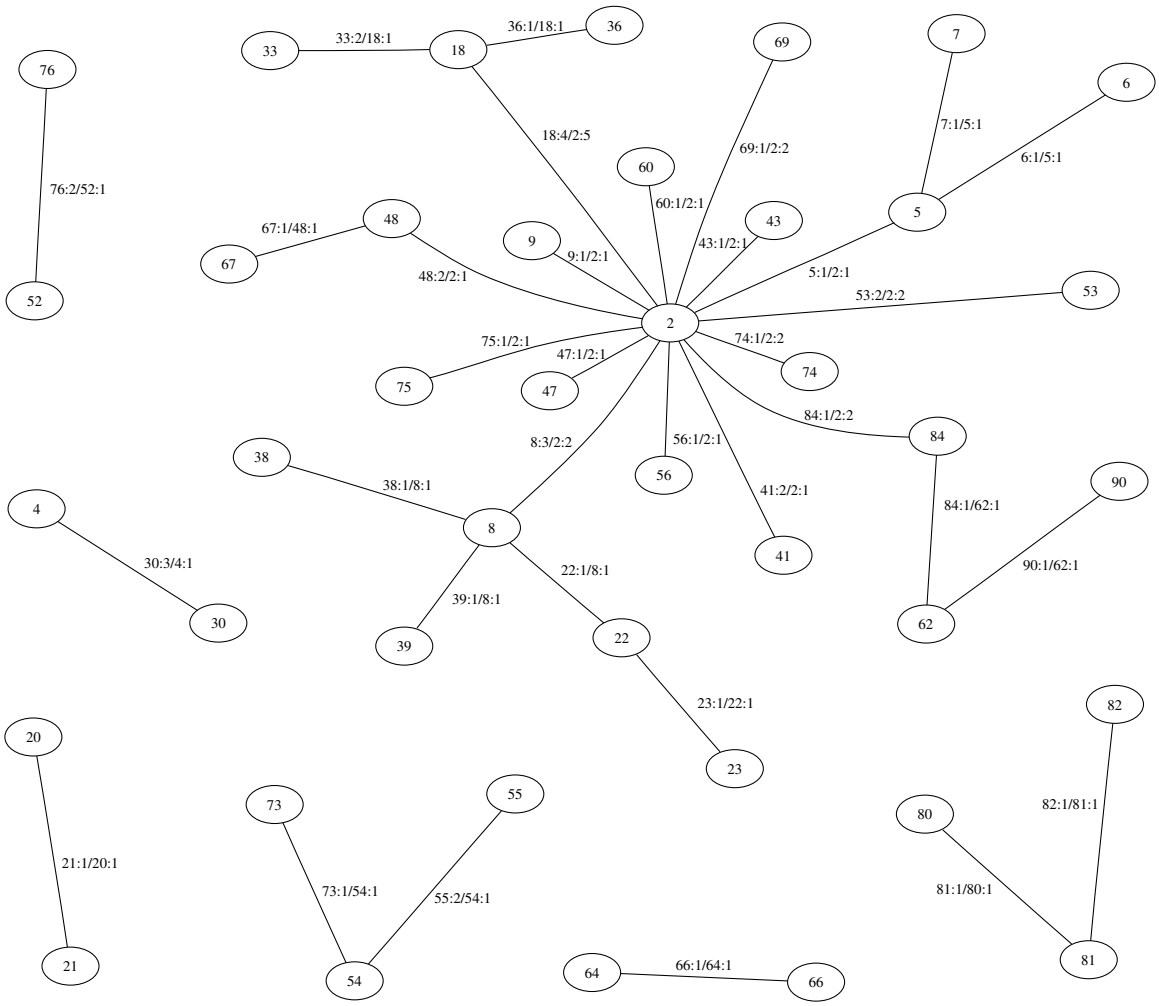


Figure 3.3: Reciprocated reply-relationships in the newsgroup `de.rec.outdoors` (Graph G_r). The figure shows relationships based on messages which were posted between May 16 and May 22. If a person a posted x messages in reply to person b, and person b posted y messages in reply to a, the edge between a and b is labeled *a:x/b:y* or vice versa. In Figures 3.3 and 3.4, the same node numbers refer to the same email address.

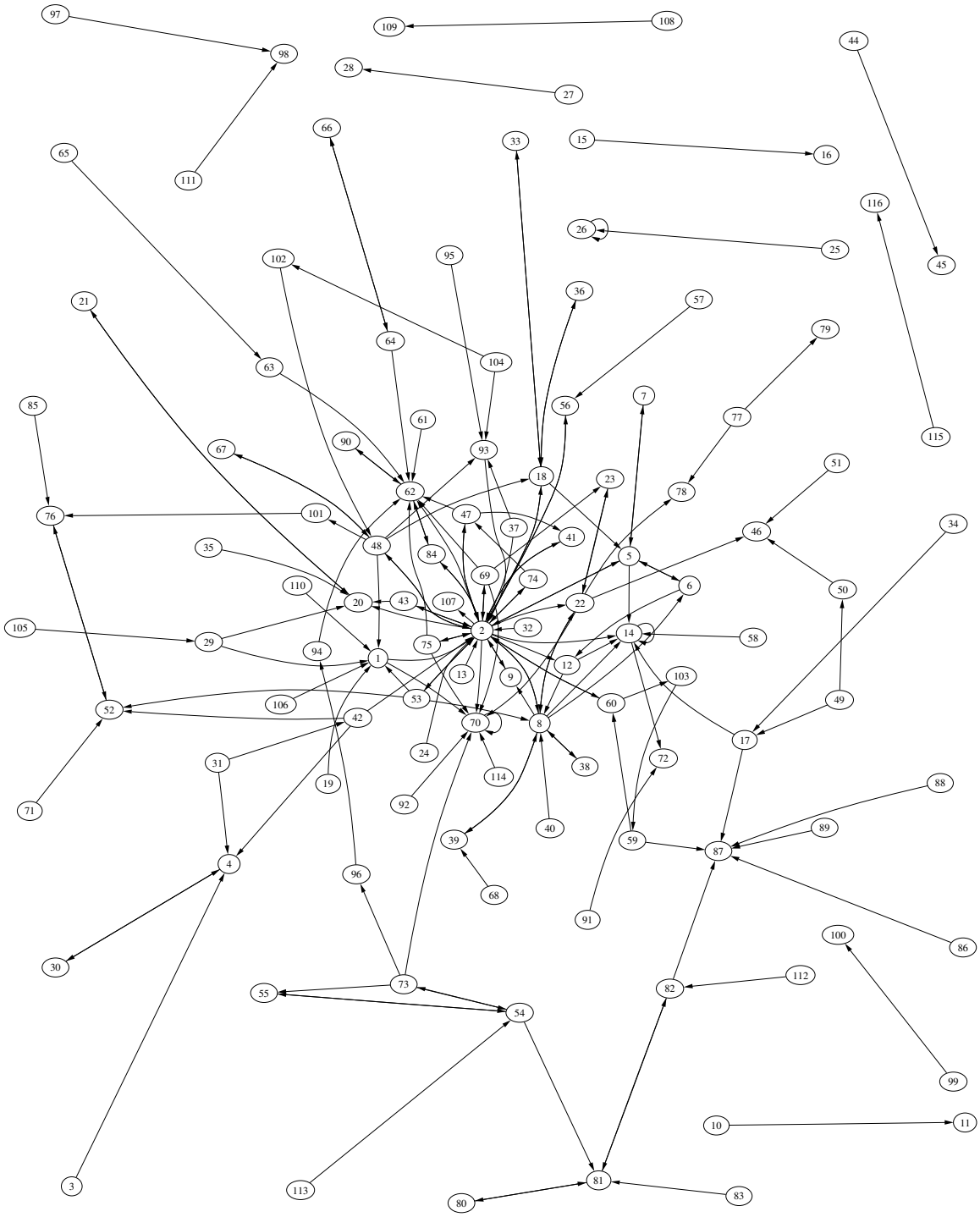


Figure 3.4: Unilateral reply-relationships in the newsgroup `de.rec.outdoors` (Graph G_d). The figure shows relationships based on messages which were posted between May 16 and May 22. Edge labels are omitted for the sake of lucidity. In Figures 3.3 and 3.4, the same node numbers refer to the same email address.

3.2.4 Instant Messaging

Instant messaging applications have gained importance both at home and at work (Lenhart et al., 2001; Cain, 2003; Jupiterresearch, 2001, 2002). Instant messaging offers near-synchronous one-to-one communication. Unlike traditional chat systems, instant messaging is based on a person-to-person communication model — at the beginning of a communication, one person opens a "connection" to another person, which may respond after the first message being sent or not. During conversations, many instant messaging clients offer functions for file exchange, initiating video/audio conferences, shared whiteboard sessions, application sharing, or starting multi user games. Instant messaging systems usually support the management of "buddylists" — i.e. the set-up of lists with persons ("buddies") the user has some relationship with. Some instant messaging clients support the management of several buddy lists depending on the kind of relationship. Most commonly, lists for family members, friends, and co-workers are distinguished. In some instant messaging systems users can object against being put on someone else's buddylist or even ignore certain users so that messages from them are blocked. Most instant messaging systems provide awareness information about buddies: next to each buddy a symbol or text is displayed indicating if the user is currently "available" (online and active), "away" (online, but not active), or "offline". In some systems, users can explicitly set their status to "invisible" (an invisible user is displayed as "offline" to other users) or to "do not disturb". In 2002, the most popular instant messaging systems were AOL AIM[11], Yahoo! Messenger[12], Trillian[13] (which has no protocol on its own but can log on all other systems in this list), ICQ Instant Message[14], MSN Messenger Service[15] (also included in Microsoft Windows XP), and AOL Instant Message[16], in this order (Jupiterresearch, 2002).

Instant messaging has been especially used by teenagers (Lenhart et al., 2001), however, studies suggest that instant messaging will increasingly gain importance in professional settings (Osterman Research, 2002; Cain, 2003). This is partly due to the informal character of instant messaging communication, as reported by Nardi et al. (2000). According to the study of Nardi et. al., the central use of instant messaging is "to support quick questions and clarifications about ongoing work tasks", "coordination and scheduling", "to coordinate impromptu social meetings", and "to keep in touch with friends and family". Extending these findings, Isaacs et al. (2002) conclude that the single-purpose character of instant messaging should not be overemphasized. Instant messaging is also used for extended work discussions being characterized by intense communication "with many short messages in a short period of time".

Although integration of instant messaging systems into traditional applications is starting to become reality (e.g., integration of MSN Messenger into Microsoft Outlook), much work still has to be done with respect to interoperability of different instant messaging systems. Currently, buddylists are only available in the system they were set up with. If two buddies use different systems, accounts for both systems are necessary — possibly even with different user names. Also integration of instant messaging with email can still be improved: although some instant messaging systems like MSN and Yahoo! automatically create an email account which can be accessed via the instant messaging client, this is not possible for arbitrary email accounts. Representing relationships based on buddylist-membership or instant messaging logs in a general framework can fill this gap and improve integration

[11] http://www.aim.com/

[12] http://messenger.yahoo.com/

[13] http://www.ceruleanstudios.com/trillian/

[14] http://www.icq.com

[15] http://messenger.msn.com/

[16] AOL Instant Message refers to the messaging service included in AOL's fee-based online network, whereas AOL AIM is the stand-alone instant messaging client.

of different instant messaging systems as well as integration of such systems into other applications.

3.3 Shared Information Spaces

Sharing and exchanging information are common tasks in cooperative work. Shared information may concern general knowledge and facts, or may document ongoing or completed group work. Two main subclasses of shared information space systems can be distinguished. The first class are hypertext systems like the world wide web. The second class contains systems supporting groups of people who share common interests or common goals ("communities") — often these systems are accessible via web interfaces and thus show hypertext features, too.

3.3.1 Hypertext Systems, Wikis, and Weblogs

Hypertext is a combination of natural language text elements and structural information allowing the interactive display of and the navigation through the text elements. Many shared information spaces consist of hypertext (Borghoff and Schlichter, 2000), since hypertext allows for more flexible references and structures (including heterogeneous data formats) than traditional linear texts. Hypertext consist of *nodes* and *links*, where nodes are small, logical information units, and links are relations between nodes, which are usually used for navigation. Nodes may be typed in order to determine the content format of the node. For example, nodes can contain plain text, markup text, graphics, video, or audio. Links between nodes may be either uni- or bidirectional. The specific semantics of each link depends on the type of the node, the type of the link, and the interpretation of the reader. Because several hypertext systems exist containing different hypertext models, the Dexter reference model was introduced in order to provide a standard for interoperability between different hypertext systems (Halasz, 1994).

Hypertext systems — especially the world wide web — are often used in research and business contexts in order to provide information about projects, organizations, or persons. Thus, the link topology of a hypertext can often yield information about social networks. For example, Contractor et al. (1998) capture knowledge-relationships between persons by analyzing links between their websites, common links to third party websites, and content analysis of the persons' websites. Kautz et al. (1997b) use lists of co-authors in technical papers and citations of papers as data sources about social networks. Xiong and Brittain (1999) as well as Minar and Donath (1999) visualize web usage by displaying a graphical map of a website indicating who is viewing which page. If the website consists of pages about persons and projects, the visualization can provide good hints about who knows who or who knows which project.

Many websites in the world wide web (still) contain static pages, which are edited offline and then transferred to a web server. However, there are systems supporting dynamic creation and editing of web pages. In the CoWebs system, for example, which was originally inspired by the Wiki project[17], every user can edit a page directly in the web browser (Dieberger and Guzdial, 2003). By simply inserting a link to a non-existing page, a new page is created and may be edited afterwards. Like reading a web page, editing a page constitutes a relationship between the person editing the web page and the original author of the page. Hypertext systems are thus sources of relationship information.

Apart from traditional web pages and Wikis, weblogs emerged as a new way of publishing information on the web in the late 1990's. The term "weblog" was coined in 1997 by Jorn Barger.[18] We-

[17] http://wiki.org

[18] http://www.robotwisdom.com/ (checked Dec. 2003)

blogs are personal webpages collecting all kinds of information their owners find interesting. Weblogs are usually regularly updated. Entries of a weblog are displayed in reverse order, i.e. the latest entry is usually displayed at the top of the page. Examples of personal weblogs are personal home pages or (sometimes categorized) link collections. Other weblogs are maintained by teams, e.g. project weblogs or news weblogs. Often, weblogs link to other weblogs, establishing a "weblog network".

The remarkable thing about weblogs is that the information they contain can be attributed to a person, or at least a group of persons. This means that reading a weblog establishes a directed relationship between the reader and the author(s) of the weblog. Similarly, linking to another weblog indicates a directed relationship between the owners of the two weblogs.

One of the reasons why weblogs have become popular is that weblogs are maintained via standard web browsers — no source editing is necessary. This makes publishing information quite easy. Weblog communities, such as http://www.blogger.com/, http://www.xanga.com/, or http://www.livejournal.com/ provide servers hosting weblog software, such that anybody can start his own weblog without needing to know anything about web page programming.

3.3.2 Supporting Virtual Communities

Computers and the Internet increasingly pervade our everyday life. More and more people "go online" for retrieving and sharing information or just communicating with friends, relatives and colleagues. Also many classical *communities* tend to (partially) move online, and even new communities evolve based on electronic communication (Rheingold, 1993; Wellman and Gulia, 1999). Communities using shared information spaces in the internet as their primary communication media are often called *virtual communities*, or, synonymously, *online communities*.

The term virtual community was coined by Rheingold (1993), who reports on his own experiences with a very early world wide virtual community, called The Well. A variety of definitions for the terms community and virtual community exist. Mynatt et al. (1997) find

> [a] loose consensus around community as referring to a multidimensional, cohesive social grouping that includes, in varying degrees: shared spatial relations, social conventions, a sense of membership and boundaries, and an ongoing rhythm of social interaction.

According to Koch (2003b), a central aspect of communities is that people voluntarily affiliate to a network, which defines some commonness between them, and which all participants benefit from. This social network forms the basis for shared activities and learning of the community.

Belonging to the same community implies influencing each other, either directly (e.g., via direct communication) or indirectly (e.g., via providing and retrieving public information). Thus, supporting virtual communities is strongly related to relationship management. While communicating with each other or providing and retrieving shared information, people constitute relationships to other people.

Classification of Communities

There are many different types of virtual communities. Carotenuto et al. (1999) distinguish between communities of interest, communities of practice, communities of purpose, and communities of passion. Communities of interest typically consist of people sharing common backgrounds or interests. Usually, such communities are not very sharply bounded. Communities of practice focus on shared professional responsibilities or activities. Common examples are programmers' communities in a large company (Vivacqua, 1999). Communities of purpose are composed of people sharing the desire to bring forward the group or organization as a whole. Knowledge management communities may be seen as examples of this type of communities. Communities of passion usually are rather small,

tightly focussed, and consist of people sharing common interests. Different from communities of interest, their members strongly identify with the topic.

Lazar and Preece (1998) classify virtual communities according to attributes, supporting software, relationship to physical communities, and boundedness. Attributes of a community may be shared goals or interests, strong ties among members, shared activities, shared resources, support among members, or social conventions. The more of these attributes a virtual community exhibits, the clearer it truly is a community (Whittaker et al., 1997). The classification by supporting software is motivated by the variety of different technologies which have been developed for supporting communities, such as Internet Relay Chat (IRC), web-based or proprietary bulletin boards (even Usenet may be regarded as a community support system), or mailing lists. Many community support systems, however, offer a combination of these services. Virtual communities can also be classified by the degree of relationship to physical communities. On the one hand, virtual communities may emerge from geographically-focused "real-life"-communities, for example, a community of citizens of a small town. On the other hand, there may be communities not related to any physical community. This type of community is especially relevant with respect to anonymity — in certain domains, members may not want their real identity to be revealed, as may be the case in communities about diseases (Leimeister et al., 2003). Another example of virtual communities not related to physical connections are communities bridging large distances, as holds for the COSMOS[19] cancer community (Leimeister et al., 2003). Communities can finally be classified according to their boundedness: in tightly bounded communities, most ties among members are within the community ("strong" ties), whereas in loosely bounded communities, members have more ties to people outside the community ("weak" ties).

Depending on the specific goal or intention, different purposes of virtual communities need to be supported. The most prominent (partly overlapping) purposes are the following.

Sharing information. The most obvious feature of many communities is the possibility for its members to share information. In the university domain, for example, shared information includes comments on courses, study groups, or events (Koch et al., 2001). In an entrepreneurship community, information about project proposals, courses, management know-how, and personal experiences may be exchanged (Schlichter et al., 2003). Personal experiences are also shared in the COSMOS cancer community (Leimeister et al., 2003), which also enables members to search for and publish professional medical information. Finally, binary data (e.g., mp3 music files) may be subject of information sharing, as is the case with peer-to-peer file-sharing software (Lechner et al., 2001).

Knowledge management. Knowledge management does not only address the management of explicit knowledge but also answering the questions "who knows what?", "who knows who?", and "who knows who knows what?" (Contractor et al., 1998). According to Wenger and Snyder (2000), communities of practice[20] are especially valuable for developing business strategies, solving problems, transferring best practices, and developing professional skills. The main approach to supporting knowledge management in communities of practice is to help people to get together and find the right persons for the problem they are trying to solve. That way, social networks make up a community's capital — the better this network is supported (including appropriate means for privacy protection), the more effective the community can be.

[19] COSMOS (http://www.cosmos-community.org) is a research project targeted at developing community support platforms especially in mobile environments. It is situated at Technische Universität München.

[20] Wenger and Snyder's definition of communities of practice also includes communities of purpose in terms of Carotenuto et al. (1999).

Supporting Awareness. To know which people are around is important for many tasks in everyday life. This includes group awareness of co-workers currently editing the same documents (Koch, 1997; Bürger, 1999) as well as awareness of friends or colleagues who are available for spontaneous communication or meetings (Nardi et al., 2000). Awareness of other people's knowledge is especially valuable for knowledge management in communities of practice.

Virtual communities are sometimes used to find other people sharing one's interests in order to initiate meetings "in real life". For example, in the role-playing community of ADRV[21], role-players can register and provide data about themselves, including their name, real-life address, contact information, the role-playing games they play, and if they would rather be game-master or player.[22] Each member of the community can query the database and look for players for the desired game, who live in the same city. Especially interesting about role playing is the fact that people *depend* on other people sharing the interest in role-playing games. However, role-playing is not too popular, and it is often difficult to find other players. Therefore, the virtual role-playing community can be very important if players are looking for other people to join their role-playing group.

Recommendation and Matchmaking. The amount of information stored in shared information spaces is often very large. Traditionally, people rely on the word of mouth when looking for reliable information or good products. In order to support this process in virtual communities, a number of recommender systems have been introduced (Resnick and Varian, 1997). These systems analyze the user's preferences, public data, and sometimes even behavior, and suggest interesting information (recommendation) or interesting persons (matchmaking). An example of a popular recommender system is the Amazon.com item recommendation system (Linden et al., 2003). Examples of the latter are Yenta (Foner, 1997) and Referral Web (Kautz et al., 1997a).

A Modular Architecture for Community Support Systems

In recent years web-based community platforms have become rather popular. Many companies recognized their value for marketing purposes, customer retention, and knowledge management. It is, however, expensive to develop and deploy virtual community platforms from scratch. Hence, generic community support systems have been developed, which can be adopted to the users' specific needs. At Technische Universität München, a java-based community support system called "Cobricks" has been developed, which follows the idea of modularization. In this section, the Cobricks community support system is briefly discussed. For a more detailed explanation of the Cobricks architecture and data concepts the reader is referred to Koch (2003b). Cobricks is also used as a testbed for the relationship management system developed in this thesis (see Sect. 7.3). At Technische Universität München, the Cobricks community support system has been deployed in several contexts, including the information system of the department of informatics[23], the MBA portal[24], the UnternehmerTUM

[21] Allgemeiner Deutscher Rollenspieler Verein (Association of German Role-Players), http://www.adrv.de

[22] A typical role-playing game needs one game-master ("storyteller" or director), and several players, who meet "in real life". The idea of the game is that each player creates a character represented by a sheet of paper with the description of the character on it, and describes the actions of the character in the story the game-master tells. These pen- and paper-based role-playing games are very similar to computer based role-playing games. In fact, many of the latter are based on rule systems originating from pen- and paper based role-playing games.

[23] http://www.in.tum.de

[24] http://portal.mba.tum.de

community[25], the COSMOS cancer community[26], and the COSMOS lifestyle community[27].

The basic model for Cobricks includes the following data concepts:

- User Profiles
- Items
- Item Annotations
- Categories
- Shared Buddylists
- Messages
- Social Relationships

These concepts can be grouped to the four main modules User Management, Item Management, Context, and Communication. Figure 3.5 depicts the basic concepts and their relations. The Social Relationship component of the User Management module is one result of this thesis and will be discussed in more detail in Sect. 7.3. In the following paragraphs the basic concepts are described briefly.

User Profiles. Each user of a Cobricks-based community support system is represented by a user profile that stores data about the user. In all cases, this data includes a user-ID which is used as a unique identifier for the user in the system. A user-ID is usually a nickname the user may choose when registering. The component providing access to the user profiles is from now on referred to as profile manager. Each user profile contains a number of attributes and their values. By specifying a user profile model for a concrete installation of Cobricks, standard attributes can be defined including types and allowed values (see Fig. 3.6). One basic design decision is to allow any Cobricks component to store arbitrary additional attributes in user profiles.

For each attribute, a user may define a set of rules which is used for access control. The concrete rule syntax depends on the rule processor chosen for the Cobricks installation. In general, possible parameters for a rule are the user-ID associated with the user profile which is accessed, and the user-ID of the user requesting access to the user profile. Based on the user profile which is accessed, the rule processor calculates the attribute value, which is then returned to the caller. For calculating the return value, the rule processor may call the profile manager and request those attributes referenced in the rule from both user profiles without applying rules (see Fig. 3.7). Thus, different values may be returned for the same attribute depending on the attributes of both users. In a mobile context, for example, a user might define a rule which returns his exact location only to members of his buddylist, and just the city to others. Any rule processor implementation must particularly take care of:

- Privacy issues. Since during rule evaluation true attribute values of both users are available, the rule syntax must ensure that return values do not allow conclusions about values of otherwise inaccessible attributes.

[25] http://www.unternehmertum.de
[26] http://www.krebsgemeinschaft.de
[27] http://www.studiosity.de

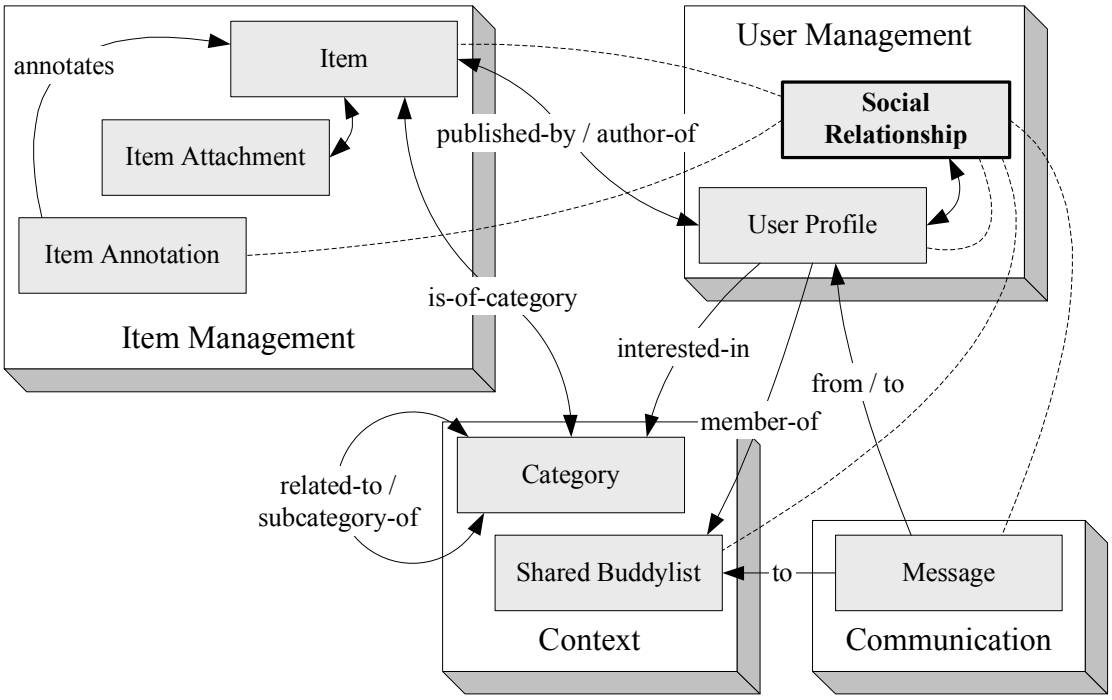


Figure 3.5: Cobricks data concepts and relations between them — according to Koch (2003b). Dashed lines indicate sources of social network data.

- Rule Priorities. The Cobricks profile manager only offers the storage and retrieval of unordered sets of rules. Any ordering of rules must be encoded in the rule syntax. Based on rule priorities, the rule processor can choose which rule to apply.

The Cobricks profile manager offers methods for managing personal buddylists. Personal buddylists are modeled as special set-valued user attributes. This allows for specifying multiple buddylists per person. Personal buddylists may be used by other Cobricks components for various purposes. The message component, for example, makes use of personal buddylists as one method for specifying the recipients of a message. In cooperation with a concrete presentation layer (see below), presence awareness information (i.e. which buddies are currently online) may be displayed to the user.

Items and Item Annotations. An item is a semi-structured data object representing an information entity. Each item possesses a unique ID, a number of basic attributes, such as item type, title, publisher, date of publication, etc., and a set of arbitrary attributes depending on the type of the item. The syntax of predefined attributes is standardized, whereas additional attributes may have arbitrary syntax (yet must be text-based — for binary data, item attachments can be used). Each item may be linked to one or more categories (see below), indicating the topic of the item or kind of information the item represents.

Each item may be annotated by one or more annotations. An item annotation is a user comment on the item. It contains an annotation ID, a title, natural language content, a (numeric) rating, the date of publication and the user-ID of the publisher. Annotations can be threaded, i.e. an annotation may either annotate an item or another annotation. Hence, item annotations are somewhat similar to hierarchic bulletin boards like Usenet, where the item forms the context for communication.

```
<userattributes>
    <attribute key="interests">
        <type multivalue="true"><![CDATA[TYPE_STRING(255)]]></type>
        <description lang="en"><![CDATA[Interests]]></description>
    </attribute>
    <branch key="basic">
        <branch key="personal">
            <attribute key="firstname" core="true">
                <type><![CDATA[TYPE_STRING(100)]]></type>
                <description lande"><![CDATA[Firstname]]></description>
            </attribute>
            <attribute key="lastname" core="true">
                <type><![CDATA[TYPE_STRING(100)]]></type>
                <description lang="en"><![CDATA[Lastname]]></description>
            </attribute>
            <attribute key="maidenname">
                <type><![CDATA[TYPE_STRING(100)]]></type>
                <description lang="en"><![CDATA[Maidenname]]></description>
            </attribute>
            <attribute key="birthday">
                <type><![CDATA[TYPE_STRING(20)]]></type>
                <description lang="en"><![CDATA[Birthday]]></description>
            </attribute>
            <attribute key="form_of_address">
                <type><![CDATA[TYPE_STRING(40)]]></type>
                <description lang="en"><![CDATA[Form of address]]></description>
                <value><![CDATA[Mrs.]]></value>
                <value><![CDATA[Mr.]]></value>
            </attribute>
            <attribute key="title">
                <type><![CDATA[TYPE_STRING(40)]]></type>
                <description lang="en"><![CDATA[Title]]></description>
            </attribute>
        </branch>
        <branch key="contact" instances="true">
            <instance key="private">
                <description lang="en"><![CDATA[private]]></description>
            </instance>
            <instance key="work">
                <description lang="en"><![CDATA[business]]></description>
            </instance>
            <branch key="online">
                <attribute key="email">
                    <type><![CDATA[TYPE_STRING(100)]]></type>
                    <description lang="en"><![CDATA[Email]]></description>
                </attribute>

                                        ⋮
```

Figure 3.6: Excerpt from a Cobricks user profile model definition

Let a, b be users, x an attribute and $R_x = \{r_{x,1}, ..., r_{x,n}\}$ the set of rules of user a for attribute x. A request getAttribute(a, x, b) of user b for attribute x of user a is evaluated as follows:

1. Profile manager gets current (true) value v of attribute x for user a from storage.
2. Profile manager invokes rule processor with parameters v, R_x, a, b.
3. Rule processor parses rules in R_x and decides which rule to apply. Let r_x^* be the rule chosen. r_x^* contains references to (other) attributes, if the access to the attribute depends on the values of those other attributes.
4. Rule processor evaluates r_x^* and uses profile manager for retrieving true attribute values (i.e. without applying rules) of users a and b, if necessary.
5. Rule processor returns v^* to profile manager.
6. Profile manager returns value v^* to caller.

Figure 3.7: Evaluation of user attributes in Cobricks

Categories and Shared Buddylists. Categories represent domain information which can be attributed to items. Categories are hierarchically arranged in a forest, where a (directed) edge from category c_1 to category c_2 indicates that c_1 is a subcategory of c_2. The subcategory-relation should be thought of as specialization — the concrete semantics depend on the design of the category forest. Figure 3.8 depicts an excerpt of the category forest of the TUM entrepreneurship community www.UnternehmerTUM.de. A second relation, referred to as "related-to", can be used to link two categories where neither of both is a subcategory of the other, but which are considered similar. For example, a category "branch←informatics←development←SAP" might be considered related to a category "branch←consulting←ERP←SAP".

Apart from these basic categories, there is a special kind of category called shared buddylists. A shared buddylist is a set of users which is referred to by the name of the shared buddylist. The intended usage of shared buddylists is for the management of small, informal groups. Every user may create a shared buddylist and grant membership to other users. Each user on a shared buddylist may in turn grant membership to other users or cancel his membership. Because shared buddylists are special categories, they can be used to attribute information items. This is particularly useful for marking items as relevant for the group. In the current category management implementation, a shared buddylist is never related to any other category (neither by the subcategory-of relation nor by the related-to relation).

The category forest of a Cobricks installation may be viewed as a strongly simplified *ontology*. Ontologies and the Semantic Web are discussed in more detail in Chapt. 4. In future versions of Cobricks, category management will probably be replaced by more powerful ontology management, which incorporates new standards like OWL[28] and RDF(S)[29].

Messages. Messages are information entities which are directed from a user or a component of the community support system (e.g. the recommendation service) to a specific group of recipients. A message consists of the following data:

- Subject

[28] Web Ontology Language (see Chapt. 4).

[29] Resource Description Framework and corresponding schema language (see Chapt. 4).

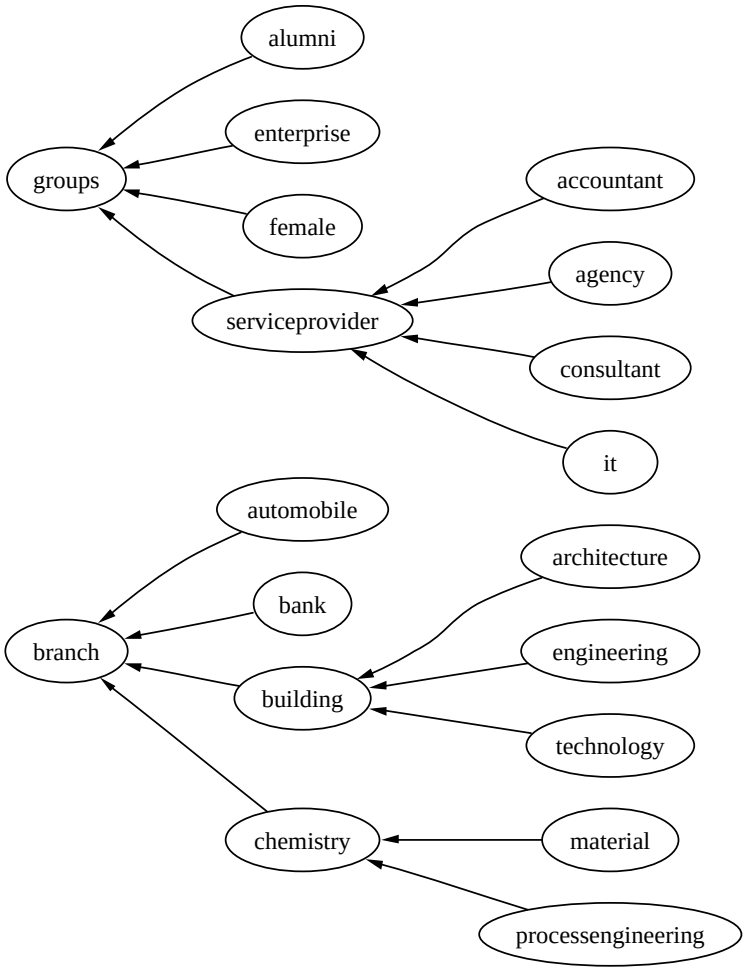


Figure 3.8: Category forest of the TUM entrepreneurship community www.UnternehmerTUM.de (excerpt)

- Message content
- Specification of recipients
- Sender of the message
- Date and time when the message was sent

Recipients can be specified by their user-ID, the name of a personal or shared buddylist, or selection conditions for the set of user profiles. The last method allows for specifying recipients according to their user profiles — for instance, all users with certain interests or all users currently located close to a special location.

Social Relationships. The social network underlying a virtual community is essential to the value of the community. Cobricks has been integrated with the InReM framework for relationship management, which is one of the results of this thesis. The InReM framework is discussed in detail in Chapt. 7. As indicated in Fig. 3.5, in Cobricks there are a variety of sources of social network data:

Messages from one user to other users. Sending a message from one member of a community to another implies relationships between sender and recipient as well as among recipients, if the message is sent to several persons at the same time. Sending a message inside a virtual community is very similar to sending an email, thus analysis patterns developed for email exchange (see Sect. 3.2.1) can be applied for internal messages with little adoption.

Membership of personal buddylists. Personal buddylists in Cobricks are very similar to buddylists in instant-messaging systems (see Sect. 3.2.4). Since Cobricks allows for multiple buddylists per user, different buddylists may be used in order to categorize buddies according to the kind of relationship with them (e.g., colleagues, family, friends). Buddylist relationships are usually directed from the owner of the buddylist to the person put on the buddylist.

Membership of shared buddylists. Similarly to personal buddylists, putting a person on a shared buddylist implies a (directed) relationship from the person granting membership to the new member of the buddylist. A shared buddylist, however, may also be viewed as a kind of mailing list. Hence, the approaches of Donath (1995) and Boyd (2002) (see Sect. 3.2.1) may be applied in order to deduce relationships among members of a shared buddylist.

Reading information items. When a person views some information which has been published by some other member of the community, a directed relationship from the reader to the publisher may be inferred. This relationship should rather not be interpreted as acquaintanceship with the publisher, but as knowledge about style and quality of that person's publications. The more different items by one publisher a reader views, the more intense is his relationship with the publisher. Although the reader gains knowledge about the content of the item, this aspect is of minor importance with respect to relationship management. Knowledge about the content does not indicate a relationship between the reader and the publisher of the item but only the reader's interest in the content of the item. Yet, the content of items published by two persons may be taken into account to derive a relationship based on shared interests between the two persons (similar to, e.g., the approach followed by Foner, 1997).

Annotating and rating items or item annotations. Publishing an annotation to an item constitutes a relationship from the annotation publisher to the item publisher. Similarly to reading information items, this indicates knowledge of the item author's style and publication quality. If the

annotation contains a rating of the item, even the annotator's assessment of the item is available. Besides annotating items, it is also possible to annotate annotations. An annotation puts strong emphasis on its author, since it usually reflects the author's personal opinion, whereas items put more emphasis on the content. Hence, the relationship deduced from annotating an item or an annotation is usually stronger than one deduced from simply viewing an item.

Viewing other persons' user profiles. A user viewing another user's profile is obviously aware of him — yet viewing a profile only once usually does not indicate a relationship. Viewing a profile repeatedly, however, often does indicate a relationship, particularly in those cases when the profile contains contact or expertise information.

Relations on user profiles. Not only interaction of users with the community support system indicate relationships with other users, but also relations on user profiles. For example, if data about the current residence is stored in the profile, a relationship may be deduced between two members living close to each other. Another example is a relationship inferred from comparing interests stored in user profiles.

Cobricks architecture overview. An installation of a Cobricks community support system consist of two layers (see Fig. 3.9). The lower layer contains the Cobricks core, which provides all management functions for the data concepts described above. The main components are Item Management, User Management, Communication, Personalization, and Context. Furthermore, import and export interfaces are provided, which can be used to synchronize data with other Cobricks installations or other proprietary systems. At the bottom of the lower layer, one might also depict a database layer. Cobricks requires an SQL database for persistent storage of community data. A special data access component provides generic access to different implementations of SQL databases and handles the peculiarities of different database systems.

The Cobricks core does not offer a built-in presentation layer — particularly there are no built-in session management or legitimization mechanisms. Instead, different implementations for the upper presentation layer of a Cobricks installation have been developed which also take care of registration and legitimization. The two most prominent are web-based presentation via Java Servlets (also known as "Portal-Toolkit") and client-based presentation via RMI/SOAP. A presentation layer building on the Cassiopeia Community Application Server Release 5.2[30] has additionally been implemented, which makes use of this server's session management and template processing capabilities.

The Cobricks core always runs on one computer and may not be distributed. Thus, a Cobricks community support system is always centralized. For user profiles, however, there are means to use external services managing user profiles via specialized interfaces as primary storage. For more information on these issues the reader is referred to Koch (2003b, 2002b).

3.4 Workflow Management

The class workflow management covers a wide range of applications. One central aspect of workflow management systems is creating business process awareness. Borghoff and Schlichter (2000) define workflow and workflow management as follows:

> "A workflow is a finite set of sequential and / or parallel activities which are triggered by events. The activities have a defined start and completion. The terms procedure, process chain or business

[30]The Cassiopeia Community Application Server Release 5.2 was developed by Cassiopeia AG, Munich, Germany, in 2002. Currently, no public resources about this product are known to the author.

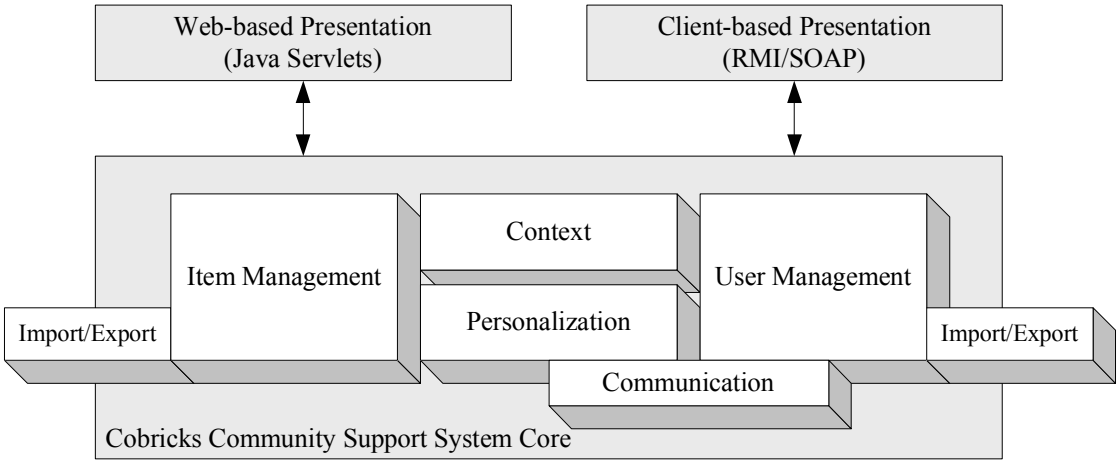


Figure 3.9: Cobricks community support system architecture overview, according to Koch (2003b)

> transaction are often used as synonyms for workflow. [...] Workflow management encompasses all the functions of modeling, specifying, simulating, analyzing, executing and monitoring a workflow."

Coordination is another central aspect of workflow management. Four different types of models for describing and defining coordination processes can be distinguished (Prinz, 1989; Teufel et al., 1995):

Procedure-oriented models. Procedure-oriented models describe a workflow as a strictly regulated process which is controlled by a central entity storing the entire information about the process.

Form-oriented models. In form-oriented models, a workflow is modeled as an "intelligent" document containing information about the route through the organization it needs to take in order to complete the business process. Form-oriented models are highly decentralized, since no central control entity is required. The document is often passed on via email. Hence these models are especially suitable for mainly serial processes.

Communication structure-oriented models. Assuming that coordination is based on communication, this group of models emphasizes communication structures in organizations. Information items are passed along communication edges connecting processing nodes. Processing nodes may be abstract organizational roles (each individual in an organization may be associated with one or several roles) or individual persons. When an information item arrives at a processing node, the node decides according to the attributes of the information item if it is passed on and which communication edge is used. Communication edges highly reflect communication structures in an organization and may also be viewed as social or organizational relationships.

Conversation-oriented models. Conversation-oriented models build on speech act theory. According to Borghoff and Schlichter (2000), "speech act theory analyzes speech as meaningful acts by communication partners in situations of shared activity." In such systems, messages can be classified according to predefined classes of speech acts, which define the intended reaction of the recipient of the message (Medina-Mora et al., 1992). An example of a speech act based system is The Coordinater (Winograd, 1988).

Workflow management systems can be regarded as important sources of social network data. Central entities of procedure-oriented models store information about message routes, which may

be used to derive social relationships between actors in the workflow. Similarly, in form-oriented models, the document itself contains routing information. Additionally, the communication system used for passing the document can be used to obtain social network data. Communication structure-oriented models obviously contain explicit social network data by defining information flow edges between processing nodes. Finally, in conversation-oriented models, speech acts can be viewed as social network data. In Action Workflow (Medina-Mora et al., 1992), for example, workflows consist of a network of atomic loops between a customer and a performer, where each loop passes through the four phases proposal, agreement, performance and satisfaction. Each such loop may be viewed as a special relationship between the customer and the performer.

The issue of social relationships in workflow management should be investigated more in depth with respect to concrete implementations. This is, however, not the primary focus of this thesis and is therefore left to future work.

3.5 Workgroup Computing

Workgroup computing includes applications for coordination support as well as for collaboration support. Examples of the former are planning systems like group calendars for supporting scheduling meetings. The latter includes decision support systems, conference management systems and all kinds of group editors. Distributed hypermedia systems — which may also be viewed as shared information spaces — can be attributed to this class, too, especially those including means for editing hypertext documents and thus serving as group editors for sets of hypertext documents.

Planning systems help to coordinate resources and capacities. With respect to social networks, planning systems are interesting since they often contain information about shared meetings or tasks. In most electronic shared calendar applications, for example, group meetings can be scheduled. For each such meeting, a list of participants is stored. Regularly attending the same meetings may be viewed as a special kind of relationship between two persons. If meetings often affect the same group of persons, this group may also be made explicit by inferring a symmetric transitive relationship between each pair of participants. In reality, however, introducing electronic shared calendars in an organization often fails, since not enough attention is paid to the balance of benefits and costs for each individual user (Grudin, 1988). Examples of systems including shared calendars are Lotus Notes[31], Microsoft Outlook[32], or Sun's CDE[33].

Shared calendars are also often used in conference management systems. Conferences may be "real" conferences with all participants in the same room or "virtual" conferences, where participants are spatially distributed. In both cases, attendance to the same conference constitutes relationships between participants.

In group editors, relationships between co-authors of documents appear. Group editors support the process of cooperative document creation. Koch (1997) defines this process as

> "joint editing of a document by multiple participants, whose shared (sub-)goal is creating the document. Because of interdependencies between individual parts of the document, the authors' activities influence each other. Thus, communication and coordination between the authors is necessary in order to achieve the common goal."[34]

[31] http://www.lotus.com

[32] http://www.microsoft.com/office/outlook/default.asp

[33] http://wwws.sun.com/software/solaris/cde/

[34] Translated by the author of this thesis.

During this process, the exchange of information about recent changes of parts of the document or about which users are currently working on the document is required. In general, supporting *awareness* about other persons' activities is a prerequisite for cooperative work in shared workspaces. Awareness is the link between group members which enables them to coordinate their contributions in an effective manner and to avoid or dissolve conflicts (Bürger, 1999; Koch, 1997). Thus, awareness is strongly related to social relationships. Obviously, co-author-relationships can be found in group editors, but there may be more detailed information available, depending on the document model of the group editor. A content analysis, for example, may yield information about the topic of the document, which might also be attributed to the relationship. Structural information may also be taken into account, for instance, if all authors contribute equally to the document, or if their sections are hierarchically organized. However, in order to explore these issues more in depth, a detailed study of the document models and cooperation and coordination protocols is necessary, which is beyond the scope of this work.

3.6 Related Work

Several existing approaches related to this work have been already mentioned in previous sections: PeCo Mediator II by Ogata et al. (2001) (see Sect. 3.2.1, p. 28), ContactMap by Nardi et al. (2002) (see Sect. 3.2.1, p. 26), Boyd's Social Network Fragments (Boyd, 2002) (see Sect. 3.2.1, p. 27), and Sack's Communication Map (Sack, 2003) (see Sect. 3.2.2, p. 32). Parts of the work of Donath et al. on visualizing conversation (Donath, 1995, 2002; Donath et al., 1999, 2001) have been outlined in Sect. 3.2.1, pp. 26 and 32.

In the remainder of this section, further approaches related to this thesis are briefly listed which have not been discussed previously.

Analysis and Exploration of Social Networks. Referral Web (Kautz et al., 1997a,b) aims at assisting people with finding trusted information on the web. Kautz et al. assume that information about trustworthiness is not publicly available on the web. Hence, such information must be gathered by finding people holding the information privately. Referral Web tries to support this process by analyzing the social network among people in a given domain (e.g., a field of research) and offering a tool which finds referral chains from one person to another. Referral Web uses co-occurrence of names in documents publicly available on the web (references on home pages, co-authorship of articles, etc.) as evidence for relationships between persons. Since only public information is used for social network analysis in order not to have to deal with privacy issues, most likely large parts of the social network keep invisible. Thus, the tool's usefulness heavily depends on people making their relationships available publicly — this may be accepted for professional and research domains, but it is certainly not for private usage of the world wide web.

Contractor et al. (1998) examine knowledge networks and how community support systems can support them. Their community support system IKNOW analyzes the social network of community members based on task and project links between them, shared skills and expertise, link topology of websites, shared links to third party websites, and content analysis of members' websites. One of the main benefits of IKNOW is effectively supporting team assembly by identifying members with shared/complementary skills.

Formalizing Trust. Abdul-Rahman and Hailes (1999) propose a model determining the trustworthiness of agents in large distributed systems. The trustworthiness of an agent depends both on personal

experiences of the agent ("direct trust") and recommendations of other agents ("recommender trust"). The recommender trust degree $t_a(x)$ of an agent a in an agent x expresses, how close most recommendations of x are to the "true" experience of a. For calculating the recommender trust degree, the direct trust degree is considered. Recommendations of agent x are then adjusted according to the recommender trust degree $t_a(x)$. After an experience of agent a with an agent b, the direct trust degree of a in b is adjusted. Furthermore, if the experience was a result of relying on a recommendation from x, the recommender trust $t_a(x)$ is adjusted (positively, if the recommendation was worse than the true experience, and negatively, otherwise). For a more detailed discussion of the algorithm and precise formulas, the reader is referred to Abdul-Rahman and Hailes (1999).

Trust in digital certificates is often established by tracing back chains of credentials (Blaze et al., 1996; Creutzig et al., 1999; Winslett et al., 2002). Credentials may be viewed as documents written in a specific credential language which are signed by the issuer by means of asymmetric encryption technologies. In order to decide whether to trust someone's credentials the credentials of the issuer can be checked, and so on, until a trusted issuer is found. Although sophisticated languages are used in such systems, credentials may simply be viewed as special relationships between the issuer and the object of the credentials. Any restrictions on or detailed description of credentials may be regarded as attributes of the relationship. Going further into detail about this topic, however, requires a discussion of the various credential languages, which is not the primary scope of this work.

Modeling Social Structures. Several approaches to formalize social structures exist. Parunak and Odell (2001) formalize social structures by using UML. Their approach particularly focuses on the definition of *roles* in multi-agent systems and is based on dependency theory and speech act theory.

Koch (2003a) proposes a formalization for virtual communities which can be used as a basis for developing community support systems. Special attention is paid to supporting the initiation phase of communities. Like the Cobricks data model, Koch's model is based on user profiles and information items. Although social relationships are not explicitly represented, Koch's analysis provides interesting measures for determining social relationships based on communication and shared interests. Therein not only explicitly provided interests are considered, but interests are also inferred from item usage patterns.

In the course of the ongoing development of the semantic web, some approaches for modeling social relationships with semantic web technologies have been proposed. Staab et al. (2001) model social relationships as RDF properties accompanied by special axioms expressing algebraic properties of these properties. A second approach is FOAF (Dumbill, 2002; RDFWeb.org, 2003), which defines a simple RDF vocabulary for expressing information about persons including very simple social relationships. The FOAF vocabulary contains the RDF property `foaf:knows`, whose domain and range are persons. The property expresses a kind of reciprocal personal acquaintance between two persons. In its current version, however, FOAF does not allow for a greater variety of relationships or structured relationships. Like this thesis, both approaches mentioned in this paragraph build upon semantic web technologies (see Chapt. 4). This work may be viewed as a generalization of those approaches.

The Stanford Framework for Interoperable Rights Management (FIRM) (Röscheisen, 1997; Röscheisen and Winograd, 1997) uses relationships for the specification of access rights. The framework is based on the assumption "that there always exists an agreement on the boundary conditions of the communication relationship" (Röscheisen, 1997, p. 15). FIRM defines how to specify these agreements, yet it is not a *language* for specifying them. Specification of access rights in FIRM is neither server-centric (like access control lists) nor client-centric (like capability lists). Instead, access rights are defined with the help of so-called communication pacts ("commpacts") representing the

relationship between the subject and the object of the access. Commpacts are results of negotiation processes. Hence, relationships in the FIRM framework are targeted towards formally describing the constituting aspects of a relationship without taking into account the social network between actors, which is the focus of this thesis. It would be interesting and promising to combine both approaches — this is one possible direction for future work.

3.7 Conclusions

This chapter has summarized the significance of social relationships in groupware and community-ware applications. Table 3.2 summarizes the kinds of relationship which have been mentioned in this chapter. Several approaches for analyzing social networks have been presented. Yet a serious drawback of all approaches is the lack of interoperability and privacy protection:

- Each system uses its own representation of social network data and its own detection mechanisms.
- Since no generally accepted data exchange format for social network data exists, it is not possible to combine different approaches in order to exchange social network data.
- Most approaches mentioned use central entities managing social network data and are thus restricted to public social network data.
- So far, privacy issues are not addressed at all — but most people do not want their social network data to be publicly available.

This thesis bridges this gap by proposing a general framework for expressing social network data which allows for interoperability as well as privacy protection. The framework builds upon technologies of the semantic web, which will be subject of the next chapter.

Kind of relationship	Sources / methods of analysis	AP	P	References
acquaintance	buddylists	as	36	
acquaintance	personal address book	as		
reciprocated acquaintance	co-occurrence of names at websites and other public documents	s	50	Kautz et al. (1997a,b); Contractor et al. (1998)
reciprocated acquaintance	FOAF-powered homepage	s	51	RDFWeb.org (2003)
acquaintance	reading public documents of an author	as	37	
acquaintance	annotating public documents of an author	as	46	
acquaintance	(shared) electronic calendars	as	49	
group membership	(shared) electronic calendars	s, t	49	
acquaintance / communication	sending an email to somebody	as	27, 28	Ogata et al. (2001); Boyd (2002)
acquaintance / communication	receiving an email from somebody	as	27, 28	Ogata et al. (2001); Boyd (2002)
acquaintance	receiving the same mail (via a “to” header or a “cc” header)	as	27, 28	Ogata et al. (2001); Boyd (2002)
acquaintance / communication	replying to a Usenet / discussion board posting	as	32	Smith (2003); Sack (2003)
communication	reciprocally sending emails	s	27, 28	Ogata et al. (2001); Boyd (2002)
communication	reciprocally replying to Usenet / discussion board postings	s	32	Smith and Fiore (2001); Sack (2003)
communication	instant-messaging each other	s	36	
group membership	subscriptions to mailing lists	s	26	Donath (1995)
communication	chatting with each other	s		
evaluation	annotating / rating public documents of an author	as		Dellarocas (2000); Kollock (1999)
evaluation	special buddylists (e.g. “friends”)	as	36	
same or similar attributes (residence, employer, interests, expertise, etc.)	comparison of user profiles	s	47	Koch (2003a) (interests)
customer-supplier	buying something at an online auction house or marketplace	as		
trust	lists of trusted keys in PGP	as, t	50	Creutzig et al. (1999)
trust	credentials	as, (t)	50	Abdul-Rahman and Hailes (1999); Blaze et al. (1996)
trust	vendor and customer rating at online auction houses or marketplaces	as, (t)		
structural / organizational relationships	workflow management systems, particularly speech acts		48	Medina-Mora et al. (1992)
collaboration	group editors, shared workspaces		49	Koch (1997)

AP: algebraic property (as: asymmetric, s: symmetric, t: transitive) P: page in this thesis

Table 3.2: Kinds of social relationships in groupware and communityware

Chapter 4

Basic Technologies of the Semantic Web

The previous chapter discussed the application area of this work — internet-based communication and shared information spaces. Especially concerning the application class "shared information spaces," both research and industry have recognized the importance of developing advanced and more flexible means for representing information and structuring information spaces. From a research perspective, one of these approaches is the semantic web. Evolving from the world wide web as it is today, the semantic web is expected to bring structure to web pages and documents and make them machine-understandable. This chapter presents an overview of the layers of the semantic web and its basic technology. It is primarily intended for readers not familiar with technologies of the semantic web in order to prepare them for understanding the formal representation of social relationships presented in Chapt. 5. Readers who are familiar with semantic web technologies may thus skip this chapter and proceed directly to Chapt. 5.

4.1 The Semantic Web Layer Cake

The world wide web (WWW) as it exists today contains huge amounts of information. 2002, there were about 3 billion static documents and over 300 million users (Patel-Schneider and Fensel, 2002). Most of that information is, however, poorly structured and written in natural language. The current WWW is addressed to *humans* — it is not machine-understandable. We experience this everyday when we use search engines. Although great improvements concerning the quality of results have been achieved in the last few years, it is still obvious that search engines do not really "understand" what we are looking for and thus have difficulties when it comes to different meanings of the same term and the context of terms. If we enter the term "jaguar", for example, a search engine is not able to find out if we are interested in the animal or the car (cf. Guha et al., 2003; Maedche et al., 2001).

The semantic web is envisioned to extend the current web and to "enable machines to comprehend semantic documents and data" (Berners-Lee et al., 2001). It is expected to make parts of the web machine-understandable, such that many tasks done by humans today can be delegated to software agents in the future (cf. Berners-Lee et al., 2001).

For the realization of this vision of the semantic web, Tim Berners-Lee proposed the semantic web layer cake (Fig. 4.1). Based on the standard technologies Unicode, URI, and XML, higher layers add more powerful languages. The remainder of this section briefly sketches these layers and afterwards discusses the RDF(S)[1] layer and the ontology layer in more detail, since these two layers are closely related to this work (see Patel-Schneider and Fensel, 2002; Fensel et al., 2003, for a more detailed description of the semantic web layer cake and related problems).

[1] RDF(S) will be used from now on to refer to both RDF and RDF Schema.

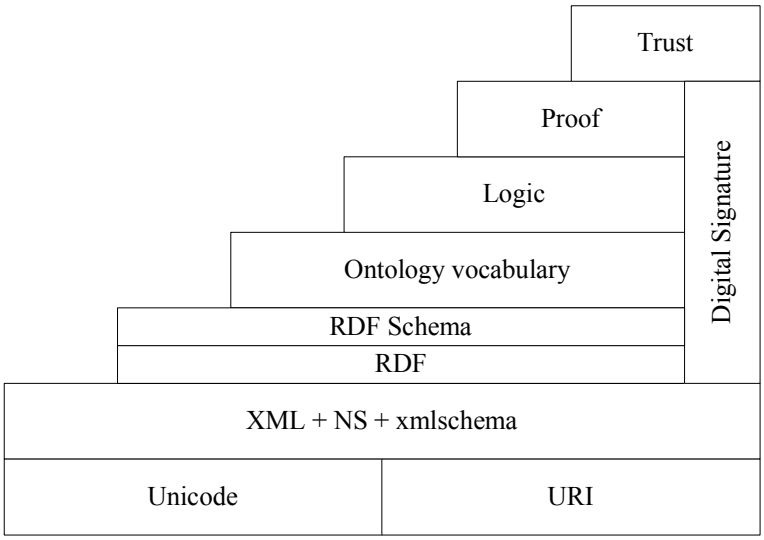


Figure 4.1: Layers of the semantic web, according to Berners-Lee (2000)

The lowest layer of the semantic web cake is provided by Unicode and URI (Berners-Lee et al., 1998). It defines standards for referencing resources on the web. URIs are generalized URLs — in contrast to a URL, which is used mainly for retrieving an object from the web, a URI simply *identifies* it.

Each URI starts with a scheme identifier. There are a variety of different URI schemes[2], e.g. http, mailto, ldap, or ftp. The scheme identifier defines the semantics of the URI and its structure.

Based upon Unicode and URI, the XML layer defines a generic representation for tree-structured documents. XML has become the standard for exchanging information on the internet. The XML namespace mechanism allows for the combination of heterogeneous XML documents and thus provides means for integrating information from various sources. XML is the syntax both for semantic web documents and for semantic annotations to these documents.

The Resource Description Framework (RDF) layer provides a mechanism for expressing statements about resources identified by URIs (Lassila and Swick, 1999). RDF is sometimes referred to as a framework for representing metadata.

The most important concept in RDF is the *statement*. A statement is a triple consisting of a subject, a predicate, and an object, and represents an assertion about the subject. A statement (s, p, o) should be interpreted as "*the property p of the resource s has the value o.*" RDF Statements are often referred to as *RDF triples*.

Although the RDF core is independent from XML, the most important way of serializing RDF data is a special XML dialect (cf. Beckett, 2003). There are, however, other forms of serializing RDF data, but they are of minor importance for this thesis and will be skipped here.

The upper half of the RDF and RDF Schema (RDF(S) from now on) layer is made up by RDF Schema (Brickley and Guha, 2003). RDF Schema allows the definition of RDF vocabularies — i.e. it provides a simple modeling language. RDF Schema is expressed in RDF itself. Its expressive power is, however, rather limited. RDF Schema introduces the concepts of classes and properties and special relations on classes and properties, i.e. a subclass-relation, a subproperty-relation, domain-restrictions

[2] http://www.iana.org/assignments/uri-schemes

of properties, and range-restrictions of properties. The benefits of RDF Schema are mainly checking RDF models for consistency or providing assistance for data input in RDF editors. For modeling the *semantics* of information, RDF Schema is by far not expressive enough. Sections 4.2 and 4.3 will discuss RDF(S) in more detail.

The ontology layer makes up the next level of the semantic web layer cake. An ontology is "a shared and common understanding of a domain that can be communicated between people and application systems" (Davies et al., 2003). An ontology defines relationships between concepts of a domain and thus formalizes certain aspects of the semantics of these concepts. Currently, the Web Ontology Language OWL is being developed (Dean and Schreiber, 2003), which uses RDF(S) as its syntax. It is supposed to be the standard ontology language for the semantic web. OWL will be the formalism chosen for representing social relationships in this work. Section 5.2 will explain in more detail, why this work chooses OWL. As OWL is the basic technology for Chapters 5 and 6, Sect. 4.3 will present a more detailed discussion of OWL.

The upper layers of the semantic web cake are still rather cloudy. The existence of a logic layer somehow suggests that there should be a richer logic for the semantic web extending the logic foundations of the ontology layer (Patel-Schneider and Fensel, 2002). The two top layers Proof and Trust are even more vague — Berners-Lee (2000) envisions mechanisms to prove the reliability and truth of assertions in the semantic web. Yet, most current efforts are concentrated on the lower levels and do not address these advanced concepts.

4.2 The Resource Description Framework

The Resource Description Framework (RDF) is the basic technology in the semantic web for representing assertions. RDF forms the syntactic basis of the Web Ontology Language (OWL). This section presents those parts of RDF which will be needed for that purpose.

"[RDF] is a foundation for processing metadata" (Lassila and Swick, 1999). It defines a standard for representing statements about resources on the web based on the four object types *resource, literal, property,* and *statement*.

Resources. The set of resources is the set of all things being described by RDF assertions. Resources may be objects on the web, such as entire web pages, email addresses, or documents, physical objects like persons or books, but also immaterial things. Anything which can be talked about can be a resource. Resources are usually referenced by URI references — there are, however, means to represent anonymous resources (see Sect. 4.2.2).

Literals. A literal is a string with an optional language identifier. A typed literal is a string accompanied by a URI referencing an XML Schema Part 2 datatype.[3] The set of literals and the set of resources are disjoint.

Properties. A property represents a certain characteristic of a resource. Properties link resources to other resources or literals. Properties are referenced by URI references. The set of properties is a subset of the set of resources — i.e. properties are entities that can be subjects of RDF statements.

[3]Language identifiers and typed literals are additional features defined by the RDF Concepts and Abstract Syntax Working Draft (Klyne and Carroll, 2003) and are not fully covered by the RDF Model and Syntax Specification (Lassila and Swick, 1999).

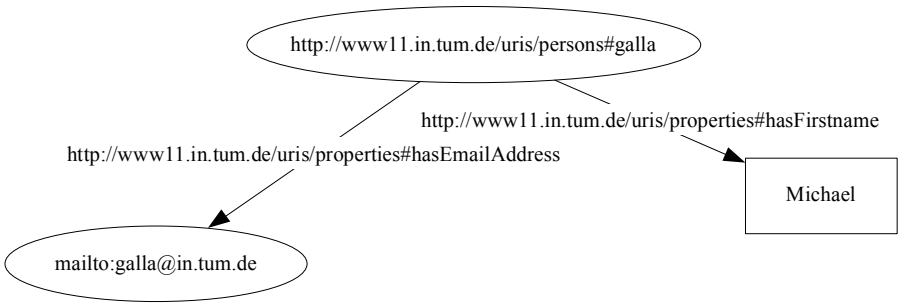


Figure 4.2: A simple RDF model. Ellipses correspond to resources, rectangles to literals.

Statements. An RDF statement consists of a reference to a resource (the *subject* of the statement), a reference to a property (the *predicate*), and either a reference to another resource or a literal (the *object*). A statement represents the assertion that the characteristic referenced by the predicate of the subject has the value referenced/represented by the object. RDF statements are sometimes referred to by the term *triple*. This work will use the notation convention $(s\,p\,o)$ to refer to an RDF statement with the subject s, the predicate p, and the object o. For each RDF statement there is a corresponding resource known as the *reification* of the statement — i.e. each statement can be subject or object of another statement. Reification is, however, of no importance for this work and is thus skipped.

The RDF model and syntax specification defines additional constructs, such as container classes, collections, and the corresponding membership properties. Since these additional features are of no relevance with respect to this work, the reader is referred to Lassila and Swick (1999) and Hayes (2003) for further details.

4.2.1 RDF Models

Later sections of this thesis will use the notion of an *RDF model*. An RDF model is a set M of RDF triples $M = \{(s\,p\,o)|s \in R, p \in P, o \in R \cup L\}$, where R is a set of resources, P is a set of properties, and L is a set of literals. Considering an OWL ontology definition, which is expressed as an RDF description, for example, the RDF model of the ontology definition consists of all triples making up the ontology definition.

RDF models are sometimes visualized as directed labeled graphs, where resources and literals are vertices, and properties are edges (such graphs are also referred to as *RDF graphs*). Figure 4.2 depicts a simple RDF model M consisting of two triples:

$$
\begin{aligned}
M = \{\quad & (\,\text{http://www11.in.tum.de/uris/persons\#galla} \\
& \text{http://www11.in.tum.de/uris/properties\#hasEmailAddress} \\
& \text{mailto:galla@in.tum.de}\,), \\
& (\,\text{http://www11.in.tum.de/uris/persons\#galla} \\
& \text{http://www11.in.tum.de/uris/properties\#hasFirstname} \\
& \text{“Michael”}\,)\,\}\,.
\end{aligned}
$$

4.2.2 Blank Nodes in RDF Graphs

Resources in an RDF graph need not necessarily be assigned an URI reference. If for any reason the URI reference of the resource is unknown or irrelevant, it may be omitted, leaving a *blank node* representing the resource. Note that for serializing RDF graphs, it may be necessary to assign local resource identifiers to resources represented by blank nodes (for further details see Beckett, 2003).

4.2.3 Typed RDF nodes

Although RDF does not contain means for defining classes, RDF defines the concept of a *class* and a typing property, known as `rdf:type`[4]. It is used in core RDF for typing resources with respect to one of the pre-defined classes `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:Statement`, `rdf:Property` and `rdf:List`. For the semantics of these pre-defined node types, the reader is referred to Lassila and Swick (1999) and Hayes (2003). The RDF Schema and OWL layers of the semantic web layer cake provide mechanisms for defining additional classes. In general, a typed resource is defined by a triple (*Resource* `rdf:type` *Class*).

4.2.4 RDF/XML Serialization

The most common serialization format for RDF is RDF/XML (Beckett, 2003).[5] This work cannot discuss RDF/XML in detail. In this subsection, only the most basic aspects of RDF/XML will be informally discussed, in order to provide a basic understanding that suffices for reading the relationship ontology definition excerpts presented in Sect. 5.3.

The RDF model depicted in Fig. 4.2 can be serialized in RDF/XML as follows.

```
<?xml version="1.0"?>
<!DOCTYPE rdf [
    <!ENTITY www11persons "http://www11.in.tum.de/uris/persons#">
    <!ENTITY www11props   "http://www11.in.tum.de/uris/properties#"> ]>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:www11persons="http://www11.in.tum.de/uris/persons#"
    xmlns:www11props="http://www11.in.tum.de/uris/properties#">

  <rdf:Description rdf:about="&www11persons;galla">
    <www11props:hasEmailAddress rdf:resource="mailto:galla@in.tum.de" />
    <www11props:hasFirstname>Michael</www11props:hasFirstname>
  </rdf:Description>

</rdf:RDF>
```

The serialization example starts with a standard XML document head defining two entities for conveniently referencing URIs of persons and properties. The `rdf:RDF` element and the associated namespace declarations form the document root of the XML document. The `rdf:Description` tag marks the beginning of a description of a resource, which is referenced by the `rdf:about` attribute. For each statement about that resource, an XML child node is defined that is named like the corresponding property — in the example above these XML nodes

[4] The RDF namespace prefix `rdf` refers to http://www.w3.org/1999/02/22-rdf-syntax-ns#.

[5] The RDF/XML serialization was originally defined by Lassila and Swick (1999). Due to certain ambiguities, the W3C is currently working on a revised version of the RDF/XML specification (Beckett, 2003).

are `www11props:hasEmailAddress` and `www11props:hasFirstname`. The first attribute value (a resource) is referenced by the `rdf:resource` attribute, the second attribute value (a literal) is the value of the XML node.

`rdf:Description` tags may be nested — i.e. it is possible to use an `rdf:Description` XML node as the value of a property. An `rdf:Description` tag may also omit the URI of the resource being described — this corresponds to blank nodes in the corresponding RDF graph. The following example illustrates nested `rdf:Description` tags with a blank node.[6]

```
<rdf:Description rdf:about="&www11persons;galla">
  <www11props:authorOf>
    <rdf:Description>
      <dc:Title>Improving Trust and Privacy in the Semantic Web
        through Identity Management
      </dc:Title>
      <dc:Subject>Semantic Web, Trust, Privacy</dc:Subject>
    </rdf:Description>
  </www11props:authorOf>
</rdf:Description>
```

RDF/XML provides a syntactic shorthand for typing resources. The RDF model

$$M = \{ \quad (\text{http://www11.in.tum.de/uris/persons\#galla} \\ \text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#type} \\ \text{http://www11.in.tum.de/projects/inrem/ns/inrem-ns\#Person}\,) \,\}$$

can be serialized either as

```
<rdf:Description rdf:about="&www11persons;galla">
  <rdf:type rdf:resource="&inrem;Person" />
</rdf:Description>
```

or as

```
<inrem:Person rdf:about="&www11persons;galla" />.
```

This abbreviated syntax is often used in later examples, because it is more compact and easier to read.

4.3 RDF Schema

The RDF Schema Working Draft (Brickley and Guha, 2003) describes how to define RDF vocabularies in terms of RDF. The central concept defined by RDF Schema is the *Class*. Classes are used to group RDF resources according to their type. The members of a class are called *instances* of the class. Note that there is a difference between the class as such and the set of its members (the *extension* of the class) — two classes may have the same extension but nevertheless be different classes. The second central concept defined by RDF Schema is a special class called `rdfs:Property`. This is the class of all resources that are RDF properties. The `rdf:type` property that has been mentioned before is a member of this class. Apart from the class `rdf:Property`, RDF Schema defines some other classes, which are, however, of minor importance for this work.

RDF Schema defines a number of members of the class `rdf:Property`, which are relevant for this work, because they are used by the Web Ontology Language (OWL). These properties are:

[6] The `dc` namespace refers to the Dublin Core Metadata Element set (see http://dublincore.org/documents/dces/).

`rdfs:range`. This property can be applied to members of `rdfs:Property` and states that the values of that property are resources of a specific class.

`rdfs:domain`. This property states that a specific property can be applied to members of certain classes only.

`rdfs:subClassOf`. The semantics of the `rdfs:subClassOf` property are: if $(a$ `rdf:type` $c_1)$, and $(c_1$ `rdfs:subClassOf` $c_2)$, then $(a$ `rdf:type` $c_2)$. The `rdfs:subClassOf` property is transitive.

`rdfs:subPropertyOf`. This property can be used to indicate that all resources which are related by a specific property are also related by a certain other property — i.e. if two resources a and b are related by a property p_1, and p_1 is a `rdfs:subPropertyOf` another property p_2, then a and b are also related by p_2. The `rdfs:subPropertyOf` is transitive.

`rdfs:comment`. The `rdfs:label` property may be used to provide a human-readable name of a resource.

`rdfs:label`. This property may be used to provide a human-readable description of a resource.

RDF Schema's expressiveness is rather limited. Apart from the `rdfs:subClassOf` property and the `rdfs:subPropertyOf` property there are no means for defining interdependencies between classes or properties. Furthermore, it is not possible to define restrictions on the membership to classes — i.e. which features a resource must exhibit in order to be a member of a specific class. RDF Schema provides means for defining RDF vocabulary, but does not allow for the definition of the *semantics* of the terms being defined. This is where OWL comes in.

4.4 The Web Ontology Language

"The Web Ontology Language OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web" (Dean and Schreiber, 2003). OWL is based on DAML+OIL[7], which is in turn a combination of the two earlier approaches DAML (Darpa Agent Markup Language)[8] and OIL (Fensel et al., 2000). Like RDF Schema, OWL defines means for specifying RDF vocabularies, but it is much more expressive. Although, according to the semantic web layer cake, ontologies should be layered on top of RDF Schema, this is not quite true with respect to OWL. More precisely, it is necessary to explicitly define *how* OWL is layered on top of RDF Schema (Patel-Schneider and Fensel, 2002). For instance, both RDF Schema and OWL define the concept of a class, but its semantics are slightly different.[9]

This work cannot give a complete introduction to OWL, its complexity classes, and how it is layered on top of RDF(S) (see Smith et al., 2003; Dean and Schreiber, 2003, for a detailed guide to OWL). Instead, it will explain central features of OWL which are important for understanding the relationship ontology vocabulary presented in Chapt. 5.

[7] http://www.daml.org/2001/03/daml+oil-index.html

[8] http://www.daml.org/

[9] The restricted versions OWL Lite and OWL DL do not allow classes to be members of other classes. RDF Schema does not impose this restriction. Hence, on the OWL Lite or OWL DL level, the semantics of OWL classes and RDF Schema classes are different. On the OWL Full level, the restriction is dropped. The class concepts of OWL Full and RDF Schema are equivalent. For the sake of compactness, this thesis will not explain these issues in more detail. The reader is referred to Dean and Schreiber (2003) for a detailed discussion of related topics.

4.4.1 OWL Syntax

OWL builds upon RDF — it uses RDF as its representation. Hence, each valid serialization method for RDF provides a serialization method for OWL. The most common serialization, which will be also used in Sect. 5.3 of this thesis, is RDF/XML.

Disregarding which serialization syntax is used, two OWL documents are equivalent if, and only if, the corresponding RDF models are equivalent.

4.4.2 OWL Class Descriptions

There are six different methods of describing an OWL class. Remark: In the OWL terminology, members of a class are sometimes called *individuals*.

Class identifier. The simplest way of defining a OWL class is to provide a URI reference for it, e.g. `<owl:Class rdf:ID="Woman" />`. This class description does not impose any restrictions on the extension of the class — it simply claims its existence.

Enumeration of members. The second way of describing a class is to explicitly enumerate all of its members:

```
<owl:Class rdf:ID="AllowedUsage">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Public"/>
    <owl:Thing rdf:about="#Anonymous"/>
    <owl:Thing rdf:about="#Private"/>
  </owl:oneOf>
</owl:Class>
```

Property restriction. The most important method of describing a class is the definition of property restrictions. This method describes restrictions on the characteristics of the members of the class. For example, a class description by property restriction is “the class of all individuals, who are female”. In RDF/XML this would be:

```
<owl:Restriction>
  <owl:onProperty rdf:about="#gender">
  <owl:hasValue rdf:about="#Female"/>
</owl:Restriction>
```

OWL defines a number of property restrictions, which cannot be discussed in detail here. The reader is referred to Smith et al. (2003) for further details.

Intersection. Classes can be defined as the intersection of several classes. If $C_1, ..., C_n$ are classes, and C is defined as the intersection of these classes, then an individual a is a member of C if, and only if, it is a member of each of the classes $C_1, ..., C_n$.

Union. Similarly, a class C can be defined as the union of several classes $C_1, ..., C_n$. In this case, C consists of all individuals, which are member of at least one of the classes $C_1, ..., C_n$.

Complement. The definition that a class C_1 is the complement of a class C_2 means that an individual is a member of C_1 if, and only if, it is not a member of C_2.

4.4.3 OWL Class Axioms

OWL introduces three constructs for expressing class axioms, namely `rdfs:subClassOf`, `owl:equivalentClass`, and `owl:disjointWith`. The meaning of the `rdfs:subClassOf` construct is essentially the same as in RDF Schema — the difference is that it is applied to OWL classes instead of RDF Schema classes. The two other constructs can be used in order to state that the class extensions of two classes consist of exactly the same members (`owl:equivalentClass`) or have no members in common (`disjointWith`).

The `rdfs:subClassOf` construct is often used for linking a named class to a class restriction:

```
<owl:Class rdf:ID="Woman">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:about="#gender">
      <owl:hasValue rdf:about="#Female"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owlClass>
```

In this example, the class `Woman` is defined as a subclass of the class of all individuals whose `gender` is `Female`. This kind of construction will be frequently used in Sect. 5.3.

4.4.4 OWL Properties and OWL Property Axioms

Similar to RDF Schema properties, OWL properties link individuals to other individuals or to literal values. Yet, OWL defines two basic property concepts: object properties and datatype properties. Entities of the former type link individuals to individuals, whereas entities of the latter type link individuals to literal values.

OWL defines a number of property axioms:

- RDF Schema constructs: `rdfs:subPropertyOf`, `rdfs:domain` and `rdfs:range`.
- Relations to other properties: `owl:equivalentProperty` and `owl:inverseOf`.
- Global cardinality constraints: `owl:FunctionalProperty` and `owl:InverseFunctionalProperty`.
- Logical property characteristics: `owl:SymmetricProperty` and `owl:TransitiveProperty`.

The RDF Schema constructs have already been explained. Their meaning in OWL is essentially the same as in RDF Schema. The axiom `owl:equivalentProperty` states that two properties have the same property extension[10], whereas `owl:inverseOf` declares an inverse relation between two properties — i.e. if p_1 and p_2 are inverse, then $(a\, p_1\, b)$ implies $(b\, p_2\, a)$ and vice versa.

Functional properties can take only one unique value for each individual they are applied to. For instance, an individual may have at most one father — the corresponding property `hasFather` should thus be declared functional. For inverse functional properties, the value of the property uniquely determines the subject. For instance, for every person, the person's father is uniquely identified — the property `isFatherOf` should thus be declared inverse functional. Symmetry and transitivity have been explained in Sect. 2.3.

[10] The *property extension* of an OWL property is the set of ordered pairs of entities linked by the property.

4.4.5 Practical OWL — Software Packages Supporting OWL

Although OWL itself is a pretty new emerging standard, its foundations are rather old. Therefore, a number of OWL Software Packages are currently being developed on the basis of older packages, or old packages are ported to OWL. The W3C maintains a list of OWL implementation projects.[11] Some of the most popular OWL-related software packages are the following.

Jena2. The Jena2 Semantic Web Framework for Java[12] has emerged from the popular Jena RDF API. In its second release, the Jena API includes ontology support for RDF Schema, DAML+OIL, and OWL.

FaCT reasoner. The FaCT reasoner[13] is a Description Logic classifier, which can be used for OWL reasoning.

Protégé-2000. Protégé-2000[14] is an ontology editor, for which an OWL export and import tool is available.

OilEd. The OilEd[15] ontology editor was originally built for OIL and DAML+OIL (Bechhofer et al., 2001). OWL support is planned for the near future. OilEd uses the FaCT reasoner for ontology reasoning.

4.5 The Semantic Web and Social Relationships

The Resource Description Framework and its extensions (RDF Schema, OWL) are a framework for describing resources and *relationships* between resources. It is quite obvious that RDF and its extensions might be used for describing social relationships between persons. The naive approach to representing social relationships in RDF is to define resources corresponding to persons and properties corresponding to relationships. This is the approach followed by FOAF (Dumbill, 2002, see also Sect. 3.6) and Staab et al. (2001).

The main drawback of this approach is that it is not possible to further explain the relationship between two persons — the only possibility is to declare a type by using a specific RDF or OWL property indicating the relationship type. RDF(S) and its extensions are not capable of representing structured relationships — strength values or further attributes of relationships like communication topics cannot be expressed.

Therefore, this work takes a different approach. Although it chooses semantic web technologies for modeling social relationships, it does not use properties for representing relationships, but individuals. This allows for arbitrary attributes of relationships. The difference between the OWL models of the two approaches is shown in Fig. 4.3.

The drawback of this approach is the loss of built-in support for symmetry and transitivity. The marriedTo relationship obviously is symmetric — if Mary is married to Peter, then Peter is, of course, married to Mary. Whereas in approach a) in Fig. 4.3, the marriedTo relationship could be represented by a symmetric OWL property, there is no corresponding construct when the relationship is represented by an individual like in approach b). Yet, the much higher expressiveness outweighs this

[11] http://www.w3.org/2001/sw/WebOnt/impls
[12] http://www.hpl.hp.com/semweb/jena2.htm, http://jena.sourceforge.net/
[13] http://www.cs.man.ac.uk/∼horrocks/FaCT/
[14] http://protege.stanford.edu/
[15] http://oiled.man.ac.uk/

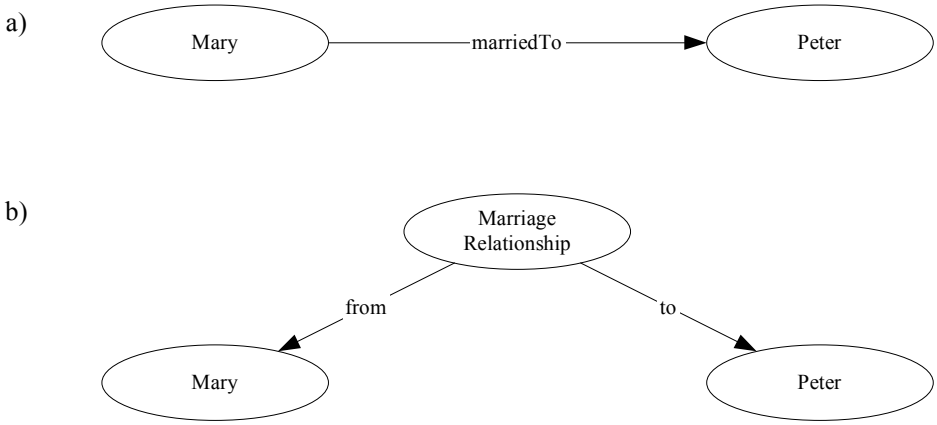


Figure 4.3: Two approaches to representing social relationship in RDF. Approach a) uses properties for representing relationship. Approach b), which this work commits to, uses resources (individuals) instead.

drawback by far. Section 5.4 explains in detail how the lack of built-in support for symmetry and transitivity can be fixed.

Chapter 5

Formalizing Relationships

A formal model of social relationships is presented, which takes into account aspects of sociology and knowledge representation. The chapter suggests forms of representation of persons, user profiles, and groups, building on the Web Ontology Language. Examples of formalizations of relationship types are presented, which are common in CSCW. After resuming detection mechanisms for relationship information, possible applications of the formalization are discussed.

5.1 Modeling Social Relationships

As has been discussed in Chapt. 3, social relationships play an important role in groupware and communityware applications. Several approaches to supporting relationship management have been sketched. Those approaches, however, have serious drawbacks. Most notably, they do not model social relationships *as such*. Instead, they take a rather statistical approach, whose primary aim is the visualization of relationship networks. Although this may be sufficient for visualization, it is certainly not for a large-scale approach to relationship management. The absence of a formal representation of relationships leads to a lack of interoperability: social network data which has been gathered by statistical procedures is useless without the knowledge of the measuring method. Therefore, such social network data can only be exchanged between systems using identical measuring methods and parameters. Yet, Chapt. 3 has pointed out that data about personal relationships is usually distributed among several applications. In order to solve the information problems discussed in Chapt. 1, all such applications must thus be considered for collecting relationship information. Furthermore, social networks consist of relationships of various types. Common tasks in social relationship management — such as exploring the own personal network or finding relationship chains to other persons — must therefore incorporate and combine relationships of various types. Existing approaches, however, fail to do so.

In this chapter, a formal representation of social relationships is suggested, which is based on semantic web technologies. This formal representation enables applications to exchange relationship information including important aspects of their meaning and algebraic properties.

In the following, the term *measuring method* denotes means for the detection of social relationships including their strength and other attributes. Measuring methods may be applied to all kinds of data (cf. Tab. 3.2) including explicitly stated relationships like formal roles or formal hierarchies, which are not represented electronically. The term *measuring method* thus subsumes all means to build a formal representation of a "real-world" relationship.

Against the background of the application area sketched in Chapters 1 and 3, the main design objectives of the formalization of social relationships are the following:

- Interoperability
- Independence from proprietary applications
- Extensibility
- Integration of privacy protection

Interoperability. Interoperability is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" (Institute of Electrical and Electronics Engineers, 1990). By employing the formalization presented in this chapter, applications should be enabled to exchange relationship information and combine it with their own data about relationships. Two dimensions of interoperability can be distinguished. First, different systems may use different internal representations of the same relationship type information. The formalization should therefore provide a common language for exchanging data about relationships. Second, different applications may use different relationship types. The formalization should thus allow for the definition of interrelations between different relationship types, such as specialization (e.g. sending emails is a special kind of communication). Furthermore, the exchange of relationship information should be situated at a high abstraction level and not depend on special communication protocols or low-level data formats.

Independence from proprietary applications. The formal representation of relationships should not depend on proprietary applications. As much as possible of the relationship's semantics should be represented, instead of an application's proprietary model of the relationship. If relationship information is to be exchanged between different applications, measuring methods for relationships must be defined globally for each relationship type. Where possible, this definition should be formal and declarative. Otherwise, natural language may be used for defining the measuring method. The claim for independence from proprietary applications leads to the need for a general representation of persons (users) and their attributes (user profiles). It should be possible to include common standards for user representation such as Microsoft Passport[1] (Microsoft Corporation, 2003) or Liberty Alliance[2] (Liberty Alliance Project, 2003). If the representation of a relationship needs to be linked to further information, such as a topic of a communication relationship, this information must be represented in an application-independent form as well. This can be achieved by applying knowledge representation technologies, such as those developed in the field of the semantic web.

Extensibility. Common relationship types, like those in Tab. 3.2, should be predefined so that applications can use these relationship types straightforward. Special organizational structures or business processes may, however, require the definition of specialized relationship types. It should be easy to publish these definitions so that other applications can make use of the specialized relationship types. It should also be possible to incorporate future standards for user (profile) representation.

Integration of privacy protection. Due to the importance of social relationships in today's business networks, relationship information must be protected against unauthorized use. At the same time, the exchange of relationship information must still be possible in order to solve the information problems discussed in Chapt. 1. Furthermore, many people simply do not want their personal data (including data about their personal social network) to be publicly available on the internet. Hence, advanced

[1] http://www.passport.net/
[2] http://www.projectliberty.org/

privacy protection mechanisms must be employed, which allow for a controlled exchange of relationship information. Since privacy protection is mainly relevant in distributed systems, these issues will be discussed in Sect. 6.5.

5.1.1 Modeling Relationship Types

According to Chapt. 2, there are different relationship types, such as individual evaluations, communication, or formal roles. As can be seen in Tab. 3.2, relationship types may be specialized with respect to aspects such as the communication media (e.g. email, usenet, etc.), or the type of evaluation (e.g. positive ratings, negative ratings, trust, distrust). The formalization of social relationships presented in this chapter thus arranges relationship types in a *specialization hierarchy*.

Similar to classes in object-oriented programming defining the semantics and the behavior of their instances, relationship types define the semantics of the corresponding relationship information. Referring to Sect. 2.7, a relationship type is defined by:

1. A globally unique identifier
2. A meaningful name
3. A natural language description
4. A set of algebraic properties
5. An optional strength attribute
6. An optional set of attribute declarations
7. An optional binary relation on pairs of user profiles
8. An optional set of references to relationship types representing generalizations

Globally unique identifier. Due to the claim for interoperability and extensibility, relationship types must be assigned a globally unique identifier. Since relationship types are resources on the internet, uniform resource identifiers (URIs) appear to be a good choice. URIs are generalizations of uniform resource locators (URLs). Different from URLs, URIs only identify objects on the web without saying anything about how to retrieve those objects (for a detailed discussion of URIs and their syntax see Berners-Lee et al., 1998).

Meaningful name. Every relationship type is assigned a meaningful name which provides a first impression of the semantics of relationships of that type. Different names can be provided for different languages. The name is intended to be used in user interfaces (e.g. for labeling edges in a graph visualization). It is not assumed to be globally unique and must thus not be used on a technical level for identifying a relationship type.

Natural language description. Like the name, the natural language description is intended for user interfaces. It clearly defines the semantics of the relationship type. This includes describing measuring methods, even if these are defined formally in a binary relation on user profiles (see below). The description of the measuring methods is necessary, because it is not sufficient to only roughly

understand the meaning of a relationship type when used in an application. Especially for implementing detection procedures it is essential to provide a precise definition of the detection algorithm reflecting the semantics of the relationship type in order to avoid misinterpretations. If the relationship type defines a profile relation (see below), the profile relation already defines the detection algorithm. Otherwise, the detection algorithm must be defined in the natural language description as precisely as possible. If the relationship type declares a strength attribute (see below), the range and semantics of the strength must be defined as well. For different languages, different versions of the description may be provided.

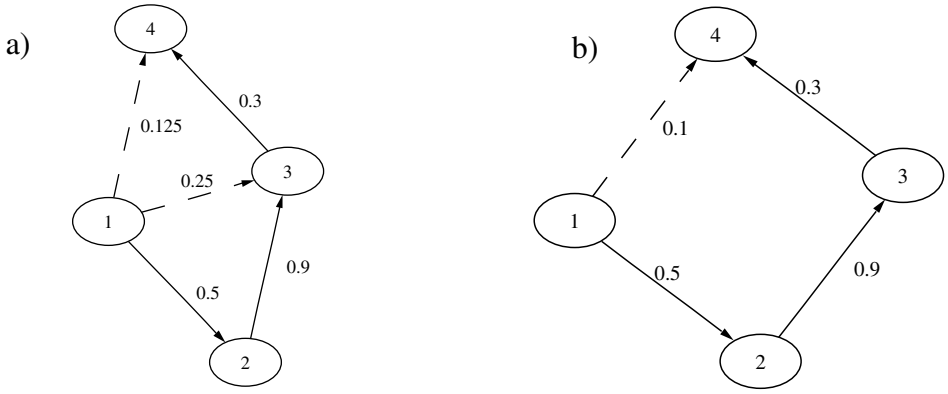
Set of algebraic properties. Referring to Sect. 2.3, a relationship type may possess a set of algebraic properties. The set of algebraic properties of a relationship type must be contradiction-free. Possible algebraic properties are symmetry, antisymmetry, asymmetry, transitivity, and non-symmetry (for definitions see Sect. 2.3). If no algebraic properties are defined, the relationship type is assumed to be non-symmetric.

The algebraic properties usually should comply with the measuring method of the relationship type: let M be the measuring method for the relationship type. Assume, the application of M to the set of data D_1 — denoted by $M(D_1)$ — yields a relationship $R(a, b)$ between Persons a and b. Then, if R is a symmetric relationship, $M(D_1)$ must also yield $R(b, a)$. If R is antisymmetric, $M(D_1)$ must not entail $R(b, a)$ unless $a = b$. If R is asymmetric, $M(D_1)$ must not entail $R(b, a)$, or $R(c, c)$ for any person c. Furthermore, let D_2 be a second set of data yielding the relationship $R(b, c)$ via the application of M. Then, if R is transitive, $M(D_1 \cup D_2)$ must entail $R(a, c)$. There are, however, some exceptions from these rules: in those cases where the underlying data is *incomplete*, algebraic properties of relationship types may be used to generate *additional knowledge*. For example, if trust relationships are assumed to be transitive, additional "trusted ties" in the social network may be inferred by calculating the transitive closure of the trust-relationship.

Strength attribute. The strength value of a relationship measures the intensity of the relationship. Common examples are the number of messages exchanged, the value of goods transferred between two persons, or a rating given to another user of an online auction house. In some cases, it is useful to normalize the strength value to the interval $[0...1]$ (e.g. for trust relationships: 0 for no trust, 1 for complete trust). The normalization of strength values allows for the comparison of the strength values of relationships of different types, which would not be possible otherwise.

Not all relationship types possess a strength value. For those relationship types the value 1 may be assumed, if the relationship between the two persons exists, and 0 otherwise. If a relationship type formally defines a strength function (see below), applications should use that definition in order to determine the strength values of relationships.

Referring to Sect. 2.1, there are various methods for assigning strength values to paths of relationships. There are, however, problems if these methods are applied to transitive relationships directly. First, neither the refined version of Peay's measure (Eqn. 2.1, p. 15) nor the refined version of Flament's measure (Eqn. 2.2, p. 15) are associative: applying the measure "step-by-step" for parts of the path first and then combining the results by applying the measure again may yield different results from applying the measure to the whole path (see Fig. 5.1). Second, if Peay's measure or Flament's measure is applied, the strength of the inferred relationship may differ from the strength value resulting from the measuring method. In the representation of the relationship it should thus be marked if the strength value was obtained by applying the transitivity property or directly via the measuring method.



In the left graph, first the strength of the relationship between 1 and 3 was calculated by using the refined version of Peay's measure (Eqn. 2.1). Afterwards, the strength of the relationship between 1 and 4 was calculated by applying the refined version of Peay's measure to the previous result. In the right graph, the strength of the relationship between 1 and 4 was calculated directly. The two methods yield different results.

Figure 5.1: The refined version of Peay's measure is not associative

Against this background, transitive relationship types which declare a strength attribute must also define the method for measuring the strength of paths. The following methods are available (cf. p. 15):

- Peay's measure: $s(p_1, ..., p_n) = \min_{i=1...n-1} s(p_i, p_{i+1})$
- Flament's measure: $s(p_1, ..., p_n) = \sum_{i=1}^{n-1} s(p_i, p_{i+1})$
- Product measure: $s(p_1, ..., p_n) = \prod_{i=1}^{n-1} s(p_i, p_{i+1})$ — This measure is especially suitable for normalized strength values.

Set of attribute declarations. A relationship type may declare additional attributes. Attributes may be single-valued or multi-valued. Examples for attributes are topics of communication relationships or shared interests (for further examples see Sect. 5.5).

Attribute values can be literal values (e.g. strings, numbers, XML), or classes/individuals of an ontology. The latter case is especially interesting, because referring to classes of an ontology allows for the inclusion of reasoning procedures. If there is a relationship with the attribute y between two persons, and y is a subclass of x, then consulting a reasoning engine yields that there also exists a relationship with attribute x between the two persons. Assume that, for example, someone is visualizing his java-programming communication network. Of course, not only communication relationships with the topic "java programming" should be visualized, but also those with the topics "java database programming", or "java swing programming". Using concepts of an ontology to attribute relationships may, however, be difficult in practice, because this requires that either all people use the same ontology, or a mapping between the ontologies exists. In the examples presented later, a list of keywords will thus be used instead of concepts (see McArthur and Bruza, 2003, for a detailed discussion about discovering connections between people by means of email analysis).

Binary relation on pairs of user profiles. By defining a binary relation on user profiles — in the following referred to as *profile relation* — it is possible to formalize the measuring method for

		Supertype				
		non-symmetric	symmetric	antisymmetric	asymmetric	transitive
Subtype	non-symmetric	yes	yes	yes	yes	yes
	symmetric	yes	yes	no	no	yes
	antisymmetric	yes	no	yes	yes*	yes
	asymmetric	yes	no	yes	yes	yes
	transitive	yes	yes	yes	yes	yes

*This is a direct consequence of the supertype being asymmetric.

Table 5.1: Allowed combination of algebraic properties of subtypes and supertypes

the relationship type. If the profile relation holds true for two persons a and b, then there exists a relationship of the respective type. The profile relation B of a relationship type representing the fact of living in the same city, for example, may be defined as

$$B(a,b) :\Longleftrightarrow \text{residence of } a \bowtie \text{residence of } b.$$

Therein, the operator $\bowtie$ assumes that its operands are classes of an ontology:

$$x \bowtie y :\Longleftrightarrow (x = y \vee x \text{ subclass of } y \vee y \text{ subclass of } x).$$

The profile relation obviously assumes a certain structure of the user profile — i.e. a profile relation can only be applied to a pair of user profiles, if both profiles comply with this structure. This issue will be discussed in more detail below.

Apart from formally defining the condition for a relationship, it makes sense to formally define the strength of a relationship. A relationship representing shared interests, for example, may define a strength measure s based on the similarity of the sets of interests of the two persons a and b:

$$s(a,b) = 2\,\frac{\text{card}(\text{interests of } a \sqcap \text{ interests of } b)}{\text{card}(\text{interests of } a) + \text{card}(\text{interests of } b)}\ ,$$

where $\text{card}(S)$ denotes the cardinality of the set S, and $S_1 \sqcap S_2$ denotes the intersection of the two sets S_1 and S_2 in consideration of subclass-relationships (see Sect. 5.1.4). For two equal sets of interest, this formula yields 1, and 0 if no shared interests exists. Again, the issue of assuming a certain structure of the user profiles is will be discussed below.

Set of generalizations. As has been mentioned above, relationship types are arranged in a hierarchy. Every relationship type may declare a list of generalizations — i.e. supertypes. Like subclasses in object-oriented programming, subtypes inherit the characteristics of the supertypes. The subtype may define additional algebraic properties and additional attributes. Algebraic properties of the subtype and all of its supertypes must be compatible: e.g. if the subtype is symmetric, then none of its generalizations may be antisymmetric (see Tab. 5.1). It is not possible for subtypes to refine the profile relation or the strength function.

5.1.2 Representing Persons

Against the background of the increasing personalization of websites, users are faced with the need to identify themselves on the internet. In most cases, the user is asked to provide a user ID which

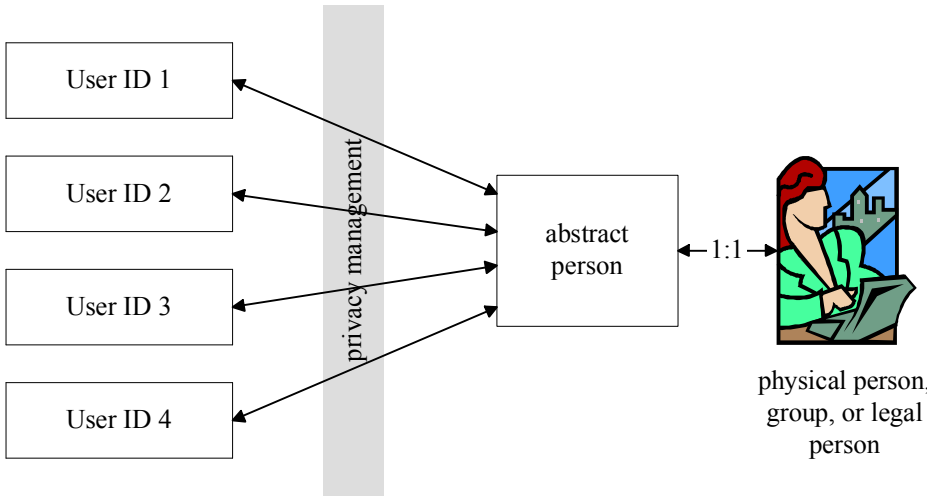


Figure 5.2: User IDs and abstract persons on the internet

uniquely identifies him with respect to the current service. Yet, when using different services, a user may have to use different user IDs: the user ID schemes of the services may vary, or a user's first choice may have been assigned to someone else already. Furthermore, a user may choose to use different user IDs due to privacy considerations (Koch, 2002b), e.g. in order to separate private accounts from business accounts. For social relationship management it is therefore necessary to distinguish between user IDs and "real" persons.

For this thesis, an *abstract person* is the abstract representation of a human being, a group or a legal person. This mapping is one-to-one — i.e. the abstract person uniquely identifies the human being, group, or legal person, and vice versa. An abstract person is uniquely referred to by one or more *user IDs* (i.e. the user IDs chosen for using services on the internet). These definitions are depicted in Fig. 5.2. Interoperable social relationship management must build on abstract persons instead of separate user IDs. In other words, different user IDs of a person must be linked to each other in such a way as to enable relationship management applications to combine relationship information associated with different user IDs of the same person (see Sect. 6.2 for more details). From now on the term "abstract" is omitted when referring to abstract persons. References to physical persons will be stated explicitly, where needed.

When talking about resources on the web, URIs are used for identification. The formalization suggested in this thesis therefore uses URIs to identify persons. Among the variety of URI schemes[3], candidates for the identification of persons are the following:

mailto scheme. The mailto URI scheme (Hoffman et al., 1998; Crocker, 1982) is the most obvious candidate for identifying persons. It describes URIs containing email-addresses (e.g. `mailto:galla@in.tum.de`). In order to allow only "reasonable" mailto-URIs, some restrictions to RFC 2368 have to be applied. First, the URI must contain exactly one email-address (RFC 2368 allows multiple addresses). Second, a complete domain specifier must be given (RFC 822 allows abbreviations if messages are sent within a domain). Third, no headers may be given in the URI apart from the `to`-header. If the `to`-header is used, it must be considered equivalent to specifying the recipient directly: `mailto:galla@in.tum.de` must be

[3] http://www.iana.org/assignments/uri-schemes

equivalent to `mailto:?to=galla@in.tum.de`. Using mailto URIs to identify persons on the internet, however, raises problems: the mailto scheme's primary purpose is to identify an email-address. Thus, if a mailto URI is used to express statements about the person associated with the email-address and about the email-address itself, conflicts or misinterpretations may occur. Assume there are two sets of RDF statements somewhere on the web, where one uses a mailto URI to identify a person (setting the value of the `rdf:type` property to `Person`) and the other one describes a property of the email-address (setting the value of the `rdf:type` property to `EmailAdress`). An reasoning engine will not be able to resolve the conflict. The resource identified by the URI may not be a person and an email-address at the same time. Against this background, mailto URIs should rather not be used to identify persons.

LDAP scheme. The LDAP URI scheme (Howes and Smith, 1997) defines how to express LDAP distinguished names and LDAP queries as URIs. The scheme is quite powerful and must be limited if used with relationship management. The subset of LDAP URIs which is suitable to identify persons is defined by the grammar `ldapuri = "ldap:///" dn`, where `dn` is an LDAP distinguished name, e.g. `cn=galla,dc=info11,dc=in,dc=tum,dc=de`. More complicated LDAP URIs are not meaningful with respect to identifying persons on the internet, because they describe actions to be performed by an LDAP server (retrieve an entry, update an entry).

http scheme. The most prominent URI scheme is the http scheme (Fielding et al., 1999). It is mainly used for retrieving web pages. As defined by RFC 2368, fragment identifiers ("#") are only allowed if the content type of the referenced document is known.[4] Additional restrictions of the scheme are not necessary. In order to avoid conflicts, a http URI used for identifying a person may not be a valid URL of a web page at the same time. The URI `http://www11.in.tum.de/persons/galla`, for example, may not be used to identify a person, because it locates the web page of this work's author.

Apart from email and news services, most services on the internet do not use URIs as user IDs. Those services should define a URI template which can be extended with the user ID. A website located at `http://www.in.tum.de/`, for example, might define the template `http://www.in.tum.de/uris/persons#ID`, where `ID` is replaced with the corresponding user ID.

5.1.3 Representing Groups

In addition to physical and legal persons, groups need to be represented in a formalization of social relationship. In a work group, for example, there is a relationship between the group leader and the group as such, which cannot be represented by a set of relationships to each individual member. Therefore, the definition of a person is inductively refined: if R is a symmetric and transitive relationship, then each equivalence class of the set of persons with respect to the relationship R is a person, too. Strictly speaking, R has to be transitive as well in order to be an equivalence relation. This can, however, be assumed without loss of generality — reflexivity is irrelevant with respect to this work, which is targeted at managing social relationships between *different* persons.

In order to uniquely identify a group, the relationship describing the group and an "anchor" (i.e. one member of the group who serves as a representative of the equivalence class) must be given.

[4] The semantics of fragment identifiers are defined according to the document type of the corresponding document and are therefore meaningless if no document is linked to the URI.

URIs for groups can be defined in two ways. The first one is to define an "artificial" URI, e.g. `http://www11.in.tum.de/uris/persons#info11`. This URI, however, does not contain any semantics of the group. The second possibility is to provide the formal representation and the anchor. In this case, a serialization of the formalization and the anchor is needed (see Sect. 5.3) along with a special URI scheme. A group URI is then defined by the following simple grammar: `"inrem:group="` `groupxml`, where `groupxml` is a URI-encoded XML serialization of the RDF model of the group.

These two possibilities to define URIs for groups reflect two slightly different qualities of groups. On the one hand, membership of a group constitutes relationships between group members (e.g. the relationship between two members of a chair). On the other hand, the group is defined based on a symmetric and transitive relationship among persons (e.g. the group of all people sharing an apartment). Hence, both ways to define group URIs are allowed in the formalization suggested in this work. The first one is more suitable for "explicit" groups (where group membership indicates relationships between group members), the second one is better for groups derived on the basis of symmetric and transitive relationships. For some groups, however, both approaches may be suitable. Hence, there may be several URIs for the same group as is the case for single persons.

5.1.4 Social Relationships Based on User Profile Comparisons

Many services on the internet gather information about their users, e.g. email addresses and other personal data. The set of all such information collected by a service with respect to a specific user ID is called the *user profile* of the specific user. Apart from the personalization of services, this information can be used to derive social relationships between two persons, e.g. living in the same city, or sharing interests. So far, most services on the internet use proprietary profile models, which are intended for internal use, only. User profiles can thus not be shared by several applications. Each application must collect the required data itself. There are, however, several approaches to deal with this problem. Most prominent, Microsoft Passport[5] (Microsoft Corporation, 2003) offers a standard for single sign-in and the sharing of profile data. A similar approach has been proposed by the Liberty Alliance Project[6] (Liberty Alliance Project, 2003), which in its current version includes identity federation and a single sign-in mechanism. Future versions are expected to include interoperability services for user profiles. Additional research has been done by Koch and Wörndl (Koch and Wörndl, 2001; Koch, 2002a,b). They propose an identity management network consisting of several so-called ID-Repositories — each storing profile data about users. Their approach includes the signature of profile data and the propagation of data between linked identities in different ID-Repositories. Section 6.2 will discuss this approach in more detail. All three approaches have in common that they aim at the separation of the user profile from the services that use it. This leads to the question how the privacy of the profile owner can be protected. This work, however, will not deal with this question in more detail.

Profile Structure

For defining profile relations it is essential that a standardized user profiles structure exists. This does not mean that it is necessary to define all possible user attributes in advance. It rather means that a standard *representation* for user profiles must be defined which the profile relation can build on. Hence, this work makes the following assumptions on how user profiles are represented:

[5] http://www.passport.com/

[6] http://www.project-liberty.org

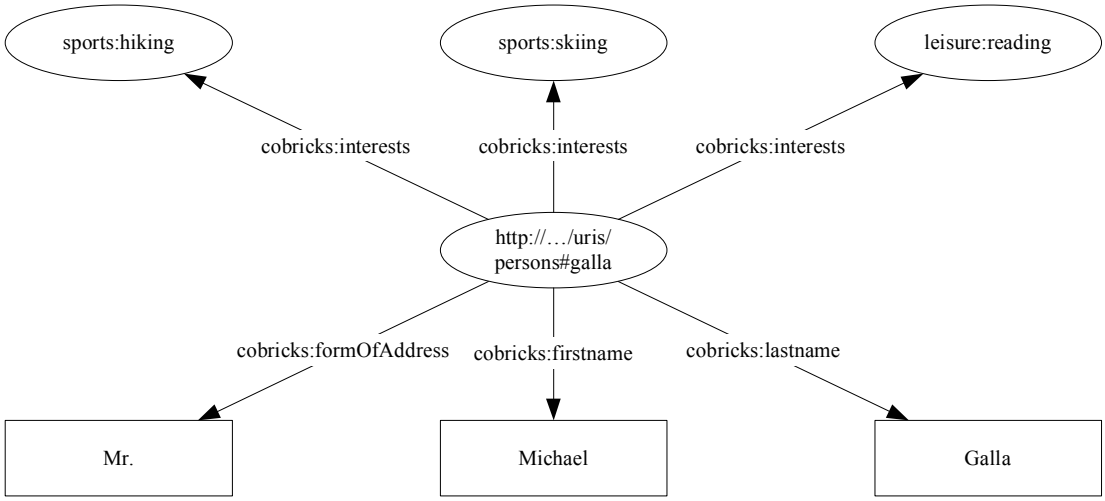


Figure 5.3: RDF graph of a Cobricks user profile based on the profile model shown in Fig. 3.6 (excerpt). Namespace prefixes (cobricks, sports, leisure) must be properly defined — the cobricks prefix is used to reference user attributes, the sports and leisure prefixes refer to ontologies about sports and leisure, respectively (of course, the sports ontology might be considered a sub-ontology of the leisure ontology, but that depends on the concrete definitions of both ontologies).

- A user profile is expressed as an RDF model.
- The user is represented by a resource, which is identified by a suitable person URI.
- User attributes correspond to OWL datatype properties, or OWL object properties.
- Attribute values may be literals (e.g. age, body height, etc.), or classes or individuals of an OWL ontology (e.g. interests, skills, etc.).

Referring to Fig. 3.6, an excerpt of the RDF model of a Cobricks user profile is depicted in Fig. 5.3. The ID-Repositories (see Sect. 6.2) used in a concrete implementation of these concepts are responsible for resolving user attribute URIs and retrieving the correct value.

Comparing User Profiles

User profiles are compared by testing if the profile relation defined in the relationship type holds true. If this is the case, a relationship of the corresponding type exists between the two persons. Apart from conjunction, disjunction, and negation, the operators defined in Tab. 5.2 are available. The set of all profile relations is defined by the grammar presented in Tab. 5.3.

As discussed in Sect.5.1, the strength of a relationship may also be formally defined. Referring to Tab. 5.3, the set of all strength functions is defined by the grammar `strengthfct = number`. As has been discussed above, it should be also stated if the strength value is normalized.

Profile-centric vs. application-centric approach

The approach of detecting relationships by comparing user profiles is quite powerful. Yet, many assumptions about the profile structure and storage must be made. A higher expressiveness requires

$\sqsubseteq$	For two classes a, b, the relation $a \sqsubseteq b$ holds true if, and only if, a is a subclass of b, or a is the same class as b.
$\sqsupseteq$	For two classes a, b, $a \sqsupseteq b$ is equivalent to $b \sqsubseteq a$.
$=$	For two classes a, b, $a = b$ holds true if, and only if, a is the same class as b.
$\bowtie$	For two classes a, b, $a \bowtie b$ holds true if, and only if, $a \sqsubseteq b$ or $b \sqsubseteq a$. For two individuals a, b, $a \bowtie b$ holds true if, and only if, $a = b$ (if a and b are classes at the same time, which is permitted in OWL full, the former definition applies). For a combination of a class and an individual, the $\bowtie$ relation never holds true (unless a reduction to one of the former cases is possible, since the individual is also a class or the class is also an individual).
$\sqcup$	For two sets of classes and individuals A, B, $A \sqcup B$ is the union of the two sets.
$\sqcap$	For two sets of classes and individuals A, B, $A \sqcap B$ is the union of the set of all individuals i where $i \in A$ and $i \in B$, the set of all individuals $j \in A$ where $\exists a \in A : j$ `rdf:type` a (analogously for the individuals in B), and the set of all classes $c \in A \sqcup B$, where $\exists a \in A$ and $b \in B : c \sqsubseteq a$ and $c \sqsubseteq b$.
`rdf:type`	For an individual a and a class c, a rdf:type c holds true if, and only if, a is an instance of c.
$\subseteq$	For two sets of classes A, B, $A \subseteq B$ holds true if, and only if, for all classes $a \in A$ there exists $b \in B : a \sqsubseteq b$ and for all individuals $i \in A$ also $i \in B$ holds true.
$\supseteq$	For two sets of classes A, B, $A \supseteq B$ holds true if, and only if, $B \subseteq A$.
$=$	For two sets of classes A, B, $A = B$ holds true if, and only if, $A \subseteq A$ and $B \subseteq A$.
any OWL object property	Let P be an object property. For two classes or individuals a, b, aPb holds true if, and only if, the corresponding triple (cf. Sect. 4.2) exists in the underlying ontology. (The operators $\sqsubseteq, \sqsupseteq, \bowtie,$ and $=$ are special object properties.)
any OWL datatype property	Let P be a datatype property. For a class or individual a and a literal value v, aPv holds true if, and only if, the corresponding triple exists in the underlying ontology.
`card` $(\cdot)$	For a set of resources A, `card` (A) denotes the cardinality of A.
$+, -, \times, /, >, <, \leq, \geq, =$	For two numbers, these mathematical operators are defined straightforward. All numbers are treated as rational numbers. The operator $=$ may also be applied to a pair of strings, as usual.
`geodist` $(\cdot, \cdot)$	For geo-coordinates c_1, c_2, `geodist` (c_1, c_2) is the shortest distance between c_1 and c_2 (rational number, in meters).
`attr` $(\cdot, \cdot)$	For $p \in \{1, 2\}$ and a user profile attribute URI u, `attr` (p, u) is the value of the attribute referenced by u of the person in position p of the relationship (a note on terminology: with respect to a relationship $R(a, b)$, a is in position 1 and b is in position 2 — i.e. R is directed from a to b). The result may be a single value or a set of values. The value must be a class or an individual.
`numattr` $(\cdot, \cdot)$	Like `attr` $(\cdot, \cdot)$, except that the (single) value must be a rational number.
`strattr` $(\cdot, \cdot)$	Like `attr` $(\cdot, \cdot)$, except that the (single) value must be a string.
`geoattr` $(\cdot, \cdot)$	Like `attr` $(\cdot, \cdot)$, except that the (single) value must be a geo-coordinate.

Class expressions such as "subclass of" or "same class as" refer to the OWL definition (Dean and Schreiber, 2003).

Table 5.2: Operators allowed in the profile relation

```
profilerelation    =  comparison / literal-comparison /
                      set-comparison / string-comparison /
                      number-comparison / "not(" profilerelation
                      ")" / "(" profilerelation ( "and" / "or" )
                      profilerelation ")"
comparison         =  resource resourceoperator resource
literal-comparison =  resource owldp owlliteral
set-comparison     =  set setoperator set
string-comparison  =  string stringoperator string
number-comparison  =  number numberoperator number
string             =  literalstring / stringattribute
number             =  literalnumber / numberattribute / geodist /
                      card / mathexpression
card               =  "card(" set ")"
set                =  attribute / "(" set "⊓" set ")" / "(" set "⊔"
                      set ")"
resource           =  attribute / uri
geodist            =  "geodist(" geocoordinate "," geocoordinate ")"
geocoordinate      =  literalgeocoordinate / geoattribute
stringattribute    =  "strattr(" position "," uri ")"
numberattribute    =  "numattr(" position "," uri ")"
geoattribute       =  "geoattr(" position "," uri ")"
attribute          =  "attr(" position "," uri ")"
position           =  "1" / "2"
mathexpression     =  "(" number mathop number ")"
resourceoperator   =  "⊑" / "⊒" / "⋈" / "=" / "rdf:type" / owlop
owlop              =  uri ; URI of an OWL object property
owldp              =  uri ; URI of an OWL datatype property
setoperator        =  "⊆" / "⊇" / "="
mathop             =  "+" / "-" / "×" / "/"
numberoperator     =  "<" / ">" / "≤" / "≥" / "="
stringoperator     =  "="
```

The definitions of `uri`, `owlliteral`, `literalstring`, `literalnumber`, and `literalgeocoordinate` (see Open GIS Consortium, Inc., 2002) are omitted, since the syntactical details are irrelevant on this conceptual level.

Table 5.3: Grammar for profile relations (on a conceptual level and in the style of Crocker and Overell, 1997)

richer profiles and thus imposes more restrictions on how user profiles are represented and accessed. Although this approach is quite feasible as far as "standard" attributes like residence, interests, and skills are concerned, it appears to be very difficult with respect to other user-related data (e.g. sending and receiving emails, group membership, etc.). In its final consequence, this approach would lead to centralized storage of all user-related data in the user profile. The user profile model would thus need to have a very high complexity in order to provide a sufficient expressiveness.

This thesis suggests a hybrid approach, allowing both methods depending on which appears convenient. In those cases mentioned above (addresses, interests, skills), the definition of reasonable profile relations is easy — assuming the existence of appropriate ontologies. Future work on user profile modeling will possibly provide new possibilities for the extension of these ideas to a higher expressiveness without loosing decentrality.

5.2 Which Formalism to Use?

The preceding section discussed how social relationships can be modeled in an internet-based context. But it has not committed to a specific formalism. Without a specific formalism it is, however, impossible to define the precise semantics of social relationships.

Reverting to the beginning of the preceding section, a number of requirements for the formal representation have been discussed:

- Interoperability
- Independence from proprietary applications
- Extensibility
- Integration of privacy protection

Relationship information is knowledge — managing relationship information is a special case of knowledge management. Hence, relationship management in an internet-context may be viewed as a special case of internet-based knowledge management. Technologies developed for knowledge management in the internet are thus likely to be good candidates for relationship management, too.

The semantic web is one of the approaches to fostering knowledge management in the world wide web. The most important component of the semantic web is a powerful ontology language which allows for the definition of extensible domain ontologies. Those approaches developed in the context of the semantic web fulfill the requirements presented above.

The two lowest levels of the semantic web layer cake (cf. Fig. 4.1) are Unicode, URI, and XML. They form the basis for a generic representation of structured data. The namespace mechanism of XML allows for the extension of XML tag vocabularies and the integration of different XML documents. On the basis of XML and URI, the RDF(S) layer defines a generic representation for objects (resources) and relationships between objects in the world wide web. In its current implementation — the Ontology Web Language (OWL) — the ontology layer provides means for describing domain vocabularies via the definition of classes, individuals, properties, and restrictions. OWL ontologies in the semantic web are extensible — ontologies can reference other ontologies and add additional knowledge — i.e. new classes, individuals, properties, or restrictions.

The OWL Reference is a W3C Candidate Recommendation and will likely be used as the standard for representing ontologies in the semantic web. This work chooses OWL as the formalism for representing knowledge about social relationships for the following reasons:

- OWL ontologies fulfill the requirements interoperability (this is one of the purposes the semantic web is designed for), independence from proprietary applications (via the generic representation of resources, individuals, properties, and restrictions), and extensibility (via namespace-mechanisms and ontology importing).
- OWL is likely to be a future standard for domain knowledge representation — tools for developing and supporting OWL ontologies are currently under development (for example, Version 2 of the Jena framework[7] includes OWL support).
- OWL is powerful enough to deal with the model for social relationships presented in the preceding section.

The following section presents an OWL ontology vocabulary for social relationships building on the Web Ontology Language OWL. Although OWL is powerful enough to define the main model for social relationships, it will be necessary to define additional semantics of algebraic properties and implications. This is partly due to the fact that relationship information is assumed to be distributed among several sources and that each source is assumed to apply privacy protection mechanisms (cf. Chapt. 6).

5.3 An OWL Vocabulary for Modeling Social Relationships

This section introduces an OWL vocabulary for the formal representation of social relationships. The central statements of Sect. 5.1 are:

- Persons are referenced by URIs. Suitable URI schemes are the http scheme and the LDAP scheme.
- Groups are equivalence classes of persons with respect to a symmetric and transitive relationship. They are defined by a representative (an “anchor”) and the relationship constituting the group.
- Groups are special persons — i.e. they may be arguments of a relationship, or members of groups.
- For each relationship type, a number of algebraic properties can be defined. Possible properties are symmetry, antisymmetry, asymmetry, transitivity, and non-symmetry.
- Each relationship type may define a strength attribute, which may be normalized. Strength values may also be obtained by the evaluation of a formally defined strength function, or by the application of the transitivity property.
- Values of attributes of a relationship can be literals, or classes and individuals of an ontology.
- A binary relation on user profiles can be defined, which formally defines the condition under which the relationship exists between two persons.
- The definition of the strength function and the profile relation is only meaningful with respect to a specific user profile model.

[7] http://www.hpl.hp.com/semweb/jena2.htm

- Types of relationship are arranged in a specialization hierarchy.

As a note on terminology, in the following, the term "class" shall refer to OWL classes, and the term "property" shall refer to OWL properties — subsuming the two subclasses "datatype property" and "object property".

For convenience and improved readability, namespace prefixes referring to the relationship ontology are omitted in the following sections. Proper namespaces must be defined for an implementation.

One of the most prominent questions when starting to design an ontology about social relationships is, whether to model relationships as classes or as properties (cf. Sect 4.5). There are good arguments for both approaches. Modeling relationships as properties allows for the use of the ontology-built-in support of algebraic properties, such as symmetry, transitivity, and inversion. The hierarchy of relationships may be expressed by defining sub-properties. There is, however, an important drawback which makes this approach insufficient. Since in a set-theoretic sense, properties are in fact binary relations, it is not possible to define further attributes for the application of a property to a pair of resources. The property either is or is not present between two resources. It is not possible to define a specific strength or annotations (i.e. attribute values). This work thus embarks on the second strategy and models relationships as classes.

The ontology presented in the remainder of this section is intended to belong to the complexity class "OWL DL". For syntactic and semantic details, the reader is referred to Dean and Schreiber (2003) and Smith et al. (2003). In the following, only fragments of the ontology will be shown. The complete ontology is presented in App. A.

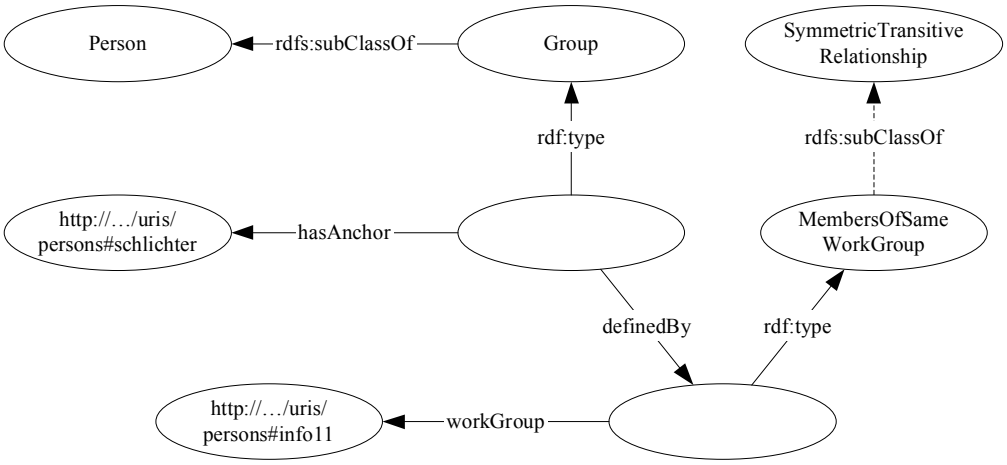
5.3.1 Persons and Groups

A person is a member of the trivial class defined by:

```
<owl:Class rdf:ID="Person" />
```

This general class does not impose any restrictions on persons, such as a specific profile model. This class may, however, be specialized in order to refer to a special user profile model. Relationships referring to special user attributes can then restrict their range to relevant subclasses of `Person`. An ID-Repository, for example, might offer its data as an OWL ontology about the user profiles stored in the repository. A reasoning engine can then combine both the relationship ontology and the profile ontology. Another specialization of the class `Person` is the class `Group`:

```
<owl:Class rdf:ID="Group">
  <rdfs:subClassOf rdf:resource="#Person" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAnchor" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#definedBy" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```



The group is defined by the symmetric and transitive relationship `MembersOfSameWorkGroup`, which describes the relationship between two members of the group (cf. Fig. 5.8). The anchor is defined by the URI reference `http://.../uris/persons#schlichter`. An additional relationship may be defined between this person and the group defined by this graph, indicating that `http://.../uris/persons#schlichter` is the leader of the work group.

Figure 5.4: RDF graph of a group definition

Therein, the two properties `hasAnchor` and `definedBy` are defined as follows:

```
<owl:ObjectProperty rdf:ID="hasAnchor">
  <rdfs:domain rdf:resource="#Group" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="definedBy">
  <rdfs:domain rdf:resource="#Group" />
  <rdfs:range rdf:resource="#SymmetricTransitiveRelationship" />
</owl:ObjectProperty>
```

The class `SymmetricTransitiveRelationship` subsumes those relationship types which are symmetric and transitive and may thus serve as an equivalence relation for persons. Figure 5.4 depicts the RDF graph of a group which is defined by the relationship representing membership of a work group. Note that the membership relationship describes the relationship between two members of the group, not between a member of the group and the group itself.

5.3.2 Types of Relationship

Types of relationship are modeled as OWL classes. The most general relationship is represented by the class `Relationship`, whose definition is shown below. The standard RDFS properties `rdfs:label` and `rdfs:comment` are used for defining the name and the description of the type, respectively. As defined by the revised RDF/XML syntax (Beckett, 2003), an `xml:lang` attribute is used to define the language of the content. It is thus possible to provide different versions of the label and the comment for different languages. Members of the class `Relationship` may be related to one Person via a `from`-property and to one Person via a `to`-property. The remaining constraints are explained in the following paragraphs.

```
<owl:Class rdf:ID="Relationship">
  <rdfs:label xml:lang="en">Generic Relationship</rdfs:label>
  <rdfs:comment xml:lang="en">
    This is the generic relationship type all other types are
    specializations of. Relationships of this type are not assumed
    to have any special meaning.
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#from" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#to" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#strength" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#normalizedStrength" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasStrengthSource" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#requiresTransitivityMeasure" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasProfileRelation" />
      <owl:maxCardinality
```

```
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTimestamp" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Because relationship types are represented by OWL classes, instances of relationships are members (individuals) of these classes. Particularly, this entails that every relationship between two persons must be viewed as a resource in the sense of RDF — i.e. it may be assigned an URI. Although this is nice, when relationships are to be referenced by other statements in the semantic web, it raises the problem of accidently producing redundant information. Assume two applications use the same set of data in order to produce relationship information, but assign different URIs to the relationship resource. Although the two representations might be equivalent except for the different URIs, any reasoning engine will assume that the two representations represent different relationships. This problem may be solved by either not assigning URIs to relationship resources, or by explicitly applying the property `owl:sameAs`: if an application finds out that two pieces of relationship information are equivalent except for their URIs (let a and b be the URIs of the corresponding relationship resources), it could explicitly add the triple $(a$ `owl:sameAs` $b)$ (or vice versa).

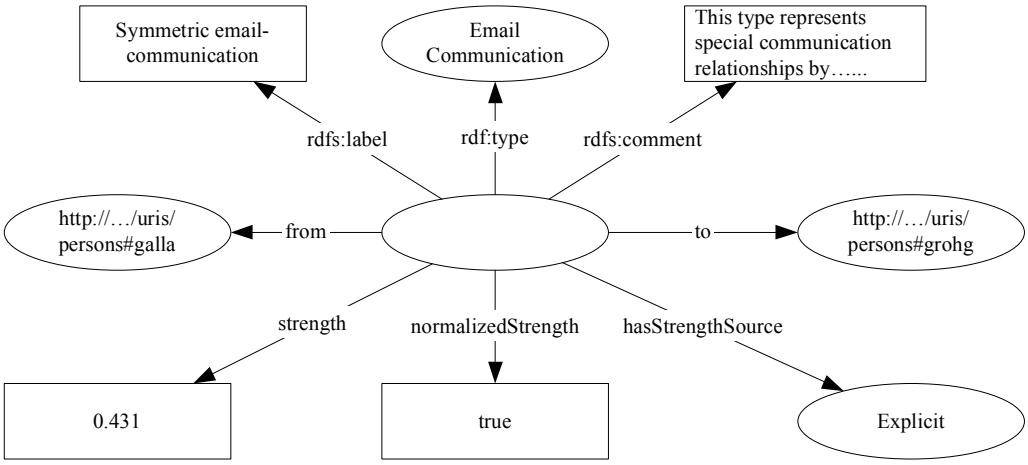
The specialization hierarchy of relationship types is expressed by applying the property `rdfs:subClassOf` to a class representing a relationship type. A subclass inherits all properties and restrictions defined for the superclass. If a special relationship type, for example, describes the relationship between a team and its leader, the range of the `to`-property may be restricted to teams, which are special persons. Any relationships that are subclasses of this type inherit this restriction.

5.3.3 Strength of Relationships and Profile Relations

The strength of a relationship is expressed by a `strength` property whose domain is a relationship, and range is an XML Schema decimal (XML Schema Datatypes are defined in Biron and Malhotra, 2001). The additional property `hasStrengthSource` indicates if the strength of the relationship was explicitly provided by the measuring method (`Explicit`), inferred by applying transitivity (members of the class `TransitivityMeasure`), or obtained by evaluation of the strength function (`StrengthFunction`). Figure 5.6 depicts the class hierarchy of possible strength sources. If the property `hasStrengthSource` is missing, `Explicit` should be assumed by applications. The property `requiresTransitivityMeasure` with domain `Relationship` and range `TransitivityMeasure` can be used in order to prescribe a specific transitivity measure to be used by applications for relationships of the corresponding type.

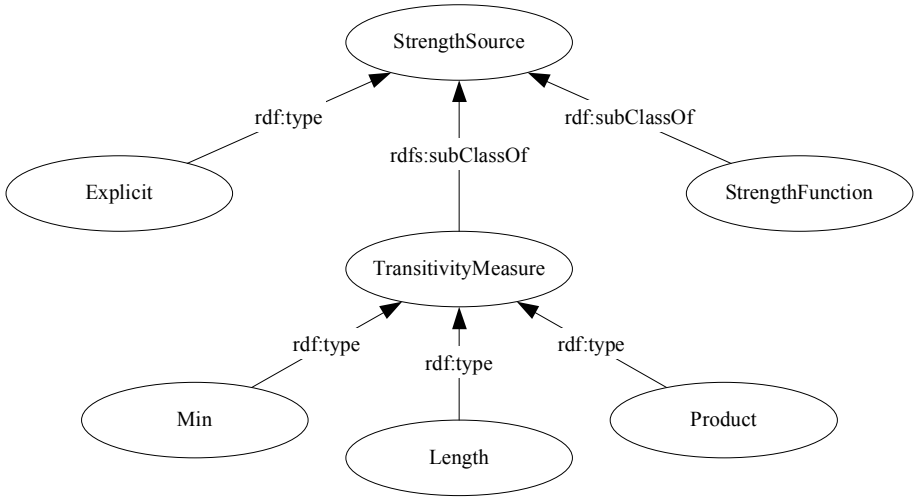
By applying the property `normalizedStrength` (range: XML Schema boolean), it is possible to define if the strength is normalized to $[0...1]$ (`true`), or not normalized (`false`). If the property `normalized` is not set, `xsd:false` should be assumed. The class `StrengthFunction` is defined as follows:

```
<owl:Class rdf:ID="StrengthFunction">
  <rdfs:subClassOf rdf:resource="#StrengthSource" />
  <rdfs:subClassOf>
```



The class `EmailCommunication` is assumed to be a subclass of the class `Communication` representing symmetric communication relationships (see also Fig. 5.8). The `rdfs:subClassOf` property linking `EmailCommunication` to `Communication` and the class `Communication` are not shown in this figure. The strength is measured according to the (normalized) formula 3.1 (p. 28). The description contains a precise definition of this formula.

Figure 5.5: RDF graph of an email-communication relationship



The three transitivity measures Min, Length, and Product refer to Peay's measure, Flament's measure, and the Product measure, respectively (see Sect. 5.1.1).

Figure 5.6: Hierarchy of methods for obtaining the strength of a relationship

```
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasStrengthDefinition" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

The profile relation of a relationship type can be defined by simply applying the string-valued property `hasProfileRelation`:

```
<owl:DatatypeProperty rdf:ID="hasProfileRelation">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

The string-valued datatype properties `hasStrengthDefinition` and `hasProfileRelation` refer to the definition of the profile relation and to the strength function, respectively (in terms of Tab. 5.3 with a suitable replacement of non-ASCII symbols).

5.3.4 Algebraic Properties

For defining algebraic properties of relationship types, the property `hasAlgebraicProperty` is provided. Its domain is `Relationship` and its range is `AlgebraicProperty`. The class `AlgebraicProperty` consists of the individuals `Symmetric`, `Antisymmetric`, `Asymmetric`, and `Transitive`. Non-symmetry does not need to be explicitly defined, since it does not impose any restrictions. Every relationship lacking an explicit definition of one or more algebraic properties is thus assumed to be non-symmetric. For conveniently defining classes possessing algebraic properties, there exists a subclass of the class `Relationship` for each property.
The class `SymmetricRelationship`, for example, is defined as follows:

```
<owl:Class rdf:ID="SymmetricRelationship">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAlgebraicProperty" />
      <owl:hasValue rdf:resource="#Symmetric" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#AntisymmetricRelationship" />
</owl:Class>
```

By defining the class to be disjoint with the class representing antisymmetric relationships, it is ensured that a class cannot at the same time be a subclass of `SymmetricRelationship` and a subclass of `AntisymmetricRelationship`. Similarly, by explicitly stating that `AsymmetricRelationship` is a subclass of `AntisymmetricRelationship`, the implication that all asymmetric relationship types are also antisymmetric (see Sect. 2.3) is accounted for.

5.3.5 Attributes

Attributes of relationships are represented by arbitrary properties. The topic of a communication relationship, for example, might be indicated by the property `hasTopic`:

```
<owl:ObjectProperty rdf:ID="hasTopic">
  <rdfs:domain rdf:resource="#CommunicationRelationship" />
</owl:ObjectProperty>

<owl:Class rdf:ID="CommunicationRelationship">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTopic">
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
      </owl:minCardinality>
    <owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Multi-valued attributes are represented by a multiple application of the corresponding property. Values of multi-valued attributes are disjunctive — i.e. in the case of two relationships with different numbers of values of the same attribute, the relationship with less values is more restricted.

Note that in the example above, omitting the range definition for the property `hasTopic` leads to the ontology being OWL Full. Hence, if a special ontology is to be used for topics, a generic class representing all possible topics should be defined in order to keep the complexity of OWL DL.

By using an OWL datatype property instead of an OWL object property, it is possible to refer to literal attribute values (e.g. XML schema strings, numbers, etc.). In particular, this allows for the definition of lists of keywords, which are not defined in an ontology.

5.3.6 Age of Relationship Information and Expiration

Social relationships are subject to temporal evolution. Relationships may fade, or change their focus, and new relationships are established. It is thus necessary to store the date and the time of measurement for each piece of relationship information. Since relationship information is to be exchanged among applications, an additional expiration date should be provided in those cases, when the expiration date of the relationship is known in advance (e.g. for formal relationships based on temporary contracts). In order to provide information about the date of detection and expiration of relationship information, the two properties `hasTimestamp` and `hasExpirationDate` can be used:

```
<owl:DatatypeProperty rdf:ID="hasTimestamp">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;datetime" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasExpirationDate">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;datetime" />
</owl:DatatypeProperty>
```

Note that according to the definition of the class `Relationship`, the timestamp is obligatory, whereas the expiration date is optional.

5.3.7 Equivalence of Relationship Information

If relationship information is exchanged between applications, it is possible that two applications possess information representing the same “real-world” relationship — this may, for example, be due

to overlapping data sets for measurement. It is thus necessary to define equivalence conditions for relationship information in order to prevent applications from storing redundant data.

As defined by the OWL Web Ontology Language Reference (Dean and Schreiber, 2003), the equivalence of two owl ontologies must be checked via the comparison of the corresponding RDF graphs instead of the serialization (because all valid methods of RDF serialization are also valid methods of OWL serialization, there are a variety of valid serializations of an OWL ontology). In fact, it is not even sufficient to compare the RDF graphs in a graph-theoretic sense. It is necessary to check the *semantic equivalence* of the two graphs. The central problem with respect to the relationship ontology presented above is to check the equivalence of two classes or individuals. More precisely, there are two complexity levels of equivalence:

Explicit Equivalence. Explicit equivalence is expressed by the `owl:sameAs`-property and can thus easily be checked. This case is especially important with respect to identity management (see Sect. 5.1.4) — the `owl:sameAs`-property may be applied by ID-Repositories (cf. Sect. 6.2) in order to state that two URIs refer to the same person. Explicit equivalence might, however, also occur in ontologies referenced by values of additional attributes.

Implicit Equivalence. This case involves all advanced concepts which may lead to two classes or individuals being equivalent (e.g. because of being subclasses of each other, or because of being subjects of an inverse-functional property, etc.).

Since the resolution of the second case involves complex reasoning algorithms, it should be delegated to an OWL reasoning engine. Explicit equivalence should, however, be dealt with by applications in any case, even if they do not contain advanced reasoning support.

5.3.8 Relationship Information Templates

For later sections of this work, it is convenient to define how to represent patterns of (incomplete) relationship information, which can be used in queries and rules. A *relationship information template* (RI template) is an incomplete formalization of a relationship, containing at least a triple $(r\ \texttt{rdf:type}\ c)$, where $c \sqsubseteq \texttt{Relationship}$. Let T be the model (i.e. the set of all triples of the corresponding RDF graph, cf. Sect. 4.2.1) of such an incomplete formalization. Let M be a (complete) model of a relationship. Let r, r' be the resources representing the relationship in T, M, respectively. Then, M *matches* T if all of the following conditions hold true.

- For each triple $(r\ P\ o) \in T$, where P is an object property and o is a class (and no individual), there exists a triple $(r'\ P\ o') \in M$, where o' is a class, and $o' \sqsubseteq o$.
- For each triple $(r\ P\ o) \in T$, where P is an object property and o is an individual, there exists a triple $(r'\ P\ o') \in M$, where o' is an individual, and $o' = o$.
- For each triple $(r\ P\ o) \in T$, where P is a datatype property (different from `strength`) and o is a literal, there exists a triple $(r'\ P\ o') \in M$, where O' is a literal, and $o' = o$.
- If there is a triple $(r\ \texttt{strength}\ s) \in T$, there is a triple $(r'\ \texttt{strength}\ s') \in M$ with $s' \geq s$ (higher strength values indicate higher strength).

A piece of relationship information matches a relationship information template, if the corresponding models match. Relationship information templates are used in access rules (cf. Sect. 6.5.2) and requests between relationship management agents (cf. Sect. 6.6).

5.3.9 Inference

OWL permits a high expressiveness of relationship ontologies. In order to make use of this high expressiveness, relationship management applications need an inference engine for answering queries to the ontology. Assume that, for instance, an ontology defines email-communication relationships to be specialized communication relationships. Both possess an attribute indicating the topics of the communication. Furthermore, assume that a relationship management application stores data about an email-communication relationship with the topic `javadatabaseprogramming` (being a subclass of `javaprogramming`). If another application (e.g. a GUI) queries the relationship management application about communication relationships about `javaprogramming`, the relationship management application can encode all relationship information it stores into an OWL ontology (as described in this section) and run an OWL inference engine on the resulting ontology. The inference engine will then yield the email-communication relationship about `javadatabaseprogramming`. Note that the subclass relationships in this example do not have to be explicitly defined, but may also be inferred by the inference engine's classification algorithm.

Since relationship types are modeled as OWL classes, OWL's built-in support for symmetric and transitive properties cannot be used with respect to relationships. Instead, applications must handle symmetry, antisymmetry, asymmetry, and transitivity themselves. This requirement will be explained in more detail in the following section.

5.4 Extension of the OWL Ontology

According to the relationship ontology suggested in Sect. 5.3, relationship types are represented by OWL classes, and relationships are represented by OWL individuals. Although this allows for the definition of the strength and additional attributes of a relationship, it causes a limitation of possible algebraic properties. Apart from the repeatedly discussed properties symmetry, antisymmetry, asymmetry, and transitivity, the algebraic properties functionality and inverse-functionality may be of interest. Furthermore, it cannot be expressed that two relationship types are inverse to each other. These properties are defined as follows:

- A binary relation R is functional, if for all x, y, z, $R(x, y) \wedge R(x, z)$ implies $y = z$.
- A binary relation R is inverse-functional, if for all x, y, z, $R(x, z) \wedge R(y, z)$ implies $x = y$.
- A binary relation R is inverse to a binary relation S, if for all x, y, $R(x, y)$ implies $S(y, x)$ and vice versa.

An example of a functional relationship type is the marriage-relationship — the same individual may only be married to one individual (at least in western societies). The has-father-relationship is an inverse-functional relationship type, since an individual may only have one father. Finally, the two relationship types parent-of and child-of are inverse to each other.

Concerning both object properties and datatype properties, OWL Full offers support for all of these algebraic properties (restricted versions are also available in OWL DL and OWL Lite). Yet, because relationship types are classes instead of properties, built-in support of OWL inference engines for these features cannot be used. Instead, the semantics of algebraic properties must be defined *outside* the relationship ontology.

In order to keep things as simple as possible, only the four algebraic properties symmetry, antisymmetry, asymmetry, and transitivity are accounted for in the relationship ontology. Relationship management applications must contain support for these properties. Concerning the additional properties

defined above, functionality and inverse-functionality mainly offer additional means for expressing equivalence, whereas the inversion-property mainly contributes to the convenience of the user. Since inversion is not included in the relationship ontology, it should be avoided to define both a relationship and its inverse — this comes down to expressing the same information in different terms and does not add any meaningful information.

The evaluation of symmetry, antisymmetry, asymmetry and transitivity is a cyclic process. The starting point of the transformation is the model M being the union of the model (i.e. the set of all triples of the corresponding RDF description, cf. Sect. 4.2.1) of the corresponding OWL relationship ontology and the set of all triples which can be generated from that ontology by an OWL reasoning engine. Then the transformation T is recursively applied to the model M, until a fixed point is reached — i.e. no additional triples can be generated by applying the transformation. Note that, because both the initial model and the transformation rules are finite, a fixed point will eventually be reached. Figure 5.7 depicts this approach. The transformation T is defined as follows:

1. Let $M' = M$.
2. Process symmetry, adding new triples to M'.
3. Process transitivity, adding new triples to M'.
4. Process antisymmetry and asymmetry, adding new triples to M'.
5. Check consistency. If the model is inconsistent, quit the transformation and report the error.
6. If $M = M'$, a fixed point is reached — in this case, quit the transformation and return the resulting model M'.

The following subsections describe the transformation procedure for each of the four algebraic properties in more detail.

5.4.1 Preliminary Remarks on the Transformation T

In order to simplify the description of the transformation, some basic notational conventions shall be introduced in this subsection. All special operators are defined in Tab. 5.2. Let M be the model of the relationship network.

Correlation. Assume r_1 and r_2 are resources representing relationships (between the same persons in the same direction) with types t_1, t_2, respectively. Then, r_1 and r_2 are *correlated* if $t_1 \bowtie t_2$, and either the more general relationship does not have any additional attributes, or both of the following conditions are met:

- For all object properties P representing additional attributes of the more general relationship type, one of the following conditions holds true.
  - There exist two triples $(r_1\, P\, o) \in M$ and $(r_2\, P\, o') \in M$, where P is an object property representing an additional attribute of the relationship, and o, o' are classes, such that $o \bowtie o'$.
  - There exist two triples $(r_1\, P\, o) \in M$ and $(r_2\, P\, o') \in M$, where P is an object property representing an additional attribute of the relationship, and o is a class, whereas o' is an individual and a member of o, or vice versa.

  - There exist two triples $(r_1\, P\, o) \in M$ and $(r_2\, P\, o) \in M$, where P is an object property representing an additional attribute of the relationship, and o is an individual.

- For all datatype properties D representing additional attributes of the more general relationship type, there exist two triples $(r_1\, D\, v) \in M$ and $(r_2\, D\, v) \in M$, where v is a literal.

Coverage. Assume r_1 and r_2 are resources representing relationships of the same type (between the same persons in the same direction). Then, r_1 is *covered* by r_2, if all of the following conditions hold true.

- For all triples $(r_1\, P\, o) \in M$, where P is an object property representing an additional attribute of the relationship, and o is a class, there exists a class o' such that $(r_2\, P\, o') \in M$ and o is a subclass of or equivalent to o', or o is a member of o' (only possible in OWL Full).
- For all triples $(r_1\, P\, o) \in M$, where P is an object property representing an additional attribute of the relationship, and o is an individual (and o is not a class at the same time, as would be possible in OWL Full), either $(r_2\, P\, o) \in M$ holds true, or there exists a triple $(r_2\, P\, o')$, where o is a member of the class o'.
- For all triples $(r_1\, D\, v) \in M$ where D is a datatype property representing an additional attribute of the relationship, $(r_2\, D\, v) \in M$ holds true.
- If the relationship type defines a strength value, the strength of r_2 is greater than or equal to the strength of r_1.

Equivalence. Assume r_1 and r_2 are resources representing relationships. Then, r_1 and r_2 are *equivalent*, if r_1 is covered by r_2 and r_2 is covered by r_1.

5.4.2 Symmetry

For symmetric relationships, the transformation is defined as follows.

1. Let r be a resource representing a symmetric relationship. Let r' be a new resource, which does not occur in M. Let R be an empty model.
2. For all properties P except `to` and `from`, and all triples $(r\, P\, o) \in M$, add $(r'\, P\, o)$ to R.
3. Assume $(r$ `from` $p_1)$ and $(r$ `to` $p_2)$. Add $(r'$ `from` $p_2)$ and $(r'$ `to` $p_1)$ to R.
4. If there exists a resource s representing a relationship which occurs in M' such that s is equivalent to r', replace the corresponding triples with those from R, setting timestamp and expiration date properly.
5. Otherwise, add all triples from R to M', setting timestamp and expiration date properly.

5.4.3 Transitivity

For two correlated transitive relationships r_1, r_2 of the types t_1, t_2, respectively, where $t_1 \bowtie t_2$, $(r_1$ `to` $p) \in M$, $(r_2$ `from` $p) \in M$, the transformation is defined as follows.

1. Let r' be a new resource, which does not occur in M'. Let R be an empty set.

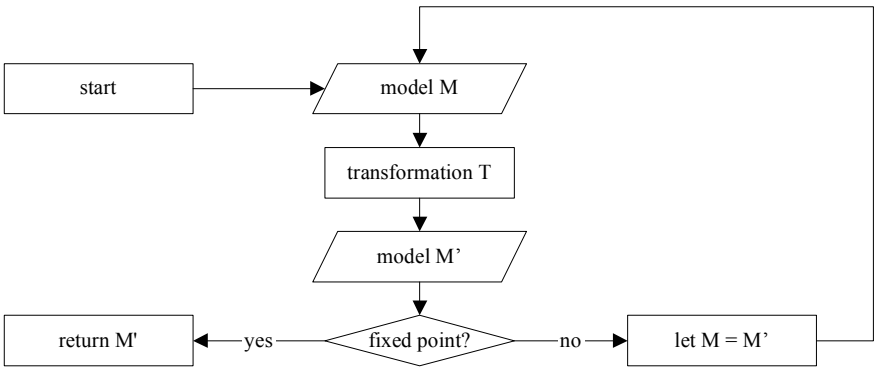


Figure 5.7: Transformation for evaluating algebraic properties of relationships

2. For all datatype properties D and all literals v such that $(r_1\, D\, v) \in M$ and $(r_2\, D\, v) \in M$, add $(r'\, D\, v)$ to R.

3. For each object property P representing an additional attribute of the more general relationship, let O_1 and O_2 be the sets of values. For each $o \in O_1 \sqcap O_2$ add the triple $(r'\, P\, o)$ to R.

4. For each remaining object property P of the more specific relationship and all values o of that property, add the triple $(r'\, P\, o)$ to R.

5. For each datatype property D representing an additional attribute of the more general relationship, let V_1 and V_2 be the sets of values. For each $v \in V_1 \cap V_2$ add the triple $(r'\, D\, v)$ to R.

6. For each remaining datatype property D of the more specific relationship and all values v of that property, add the triple $(r'\, D\, v)$ to R.

7. Assume $(r_1$ `from` $p_1)$ and $(r_2$ `to` $p_2)$. Add $(r'$ `from` $p_1)$ and $(r'$ `to` $p_2)$ to R.

8. If there exists a resource s representing a relationship and occurring in M' such that s is equivalent to r', replace the corresponding triples with those from R, setting strength-related properties, timestamp and expiration date properly.

9. Otherwise, add all triples from R to M', setting strength-related properties, timestamp and expiration date properly.

5.4.4 Antisymmetry and Asymmetry

Let r_1, r_2 be correlated resources, representing antisymmetric relationships. If $(r_1$ `from` $p_1) \in M$, $(r_1$ `to` $p_2) \in M$, $(r_2$ `from` $p_2) \in M$, and $(r_2$ `to` $p_1) \in M$, add $(p_1$ `owl:sameAs` $p_2)$ to M'.

If both relationships are antisymmetric, and the conditions above apply, the model is inconsistent.

5.5 Basic Types of Relationship

The previous sections have presented a formalization of social relationships in form of an OWL ontology. This section gives examples of relationship types, which can be represented using that formalization. This list of types is not exhaustive. Still, it provides an insight into how frequent relationship types can be properly modeled using the previously discussed modeling primitives. Figure 5.8 depicts the hierarchy of common relationship types which are described in the following list.

Acquaintance. This general relationship type represents the fact that a person knows the other. It does neither imply any evaluations of the other person, nor a face-to-face contact between the two persons.

PositiveEvaluation. *PositiveEvaluation* is a subtype of *Acquaintance*. It reflects a kind of friendly attitude towards the target of the relationship, e.g. good experiences, fondness, or trust.

BuddylistMembership. This type represents relationships based on membership of the target person on a person's buddylist. It is a subtype of *PositiveEvaluation* and has one attribute referring to the list the target is a member of, e.g. "friends", "family", or "co-workers".

Trust. For expressing a trust-relationship from one person to another, this type can be used. It indicates general trust and is a subtype of *PositiveEvaluation*. A strength value must be provided, ranging from 0 (no trust, equivalent to the absence of the relationship) to 1 (complete trust). Note that this type is not transitive. Special subtypes should be defined with respect to more specific situations, e.g. *RecommenderTrust* (see below).

RecommenderTrust. These relationships are specialized trust relationships. They indicate that a person trusts the other person's recommendations, e.g. product ratings, or vendor ratings. Note that this type is transitive. For inferring the strength of relationship paths, the product measure is to be used. Since strength values are normalized, this implies that the strength of relationship chains decreases with increasing length (except if all relationships have strength 1).

NegativeEvaluation, BlacklistMembership, Distrust, RecommenderDistrust. These types are analogs to *PositiveEvaluation* and the corresponding subtypes. They express, however, negative attitudes. (It is not true that a trust relationship of strength 0 indicates a distrust relationship of strength > 0. Trust and distrust relationships of low strength indicate neutral evaluation.)

Communication. Relationships of this type represent all kinds of communication. Communication relationships are assumed to be symmetric. The attribute "topic" indicates the topics of the relationship as a set of literals.

EmailCommunication. This type specializes the previous type by specifying email as the medium of the communication. Relationships of this type possess a normalized strength value as defined by formula 3.1 (p. 28). The RDF graph of a relationship of this type is depicted in Fig. 5.5 (topics are omitted in the graph).

Transaction. *Transaction* relationships subsume all relationships consisting in the transfer or exchange of goods or money. Subtypes of this type should be defined in order to specialize the kind of transaction, e.g. *BuysFrom* (see below).

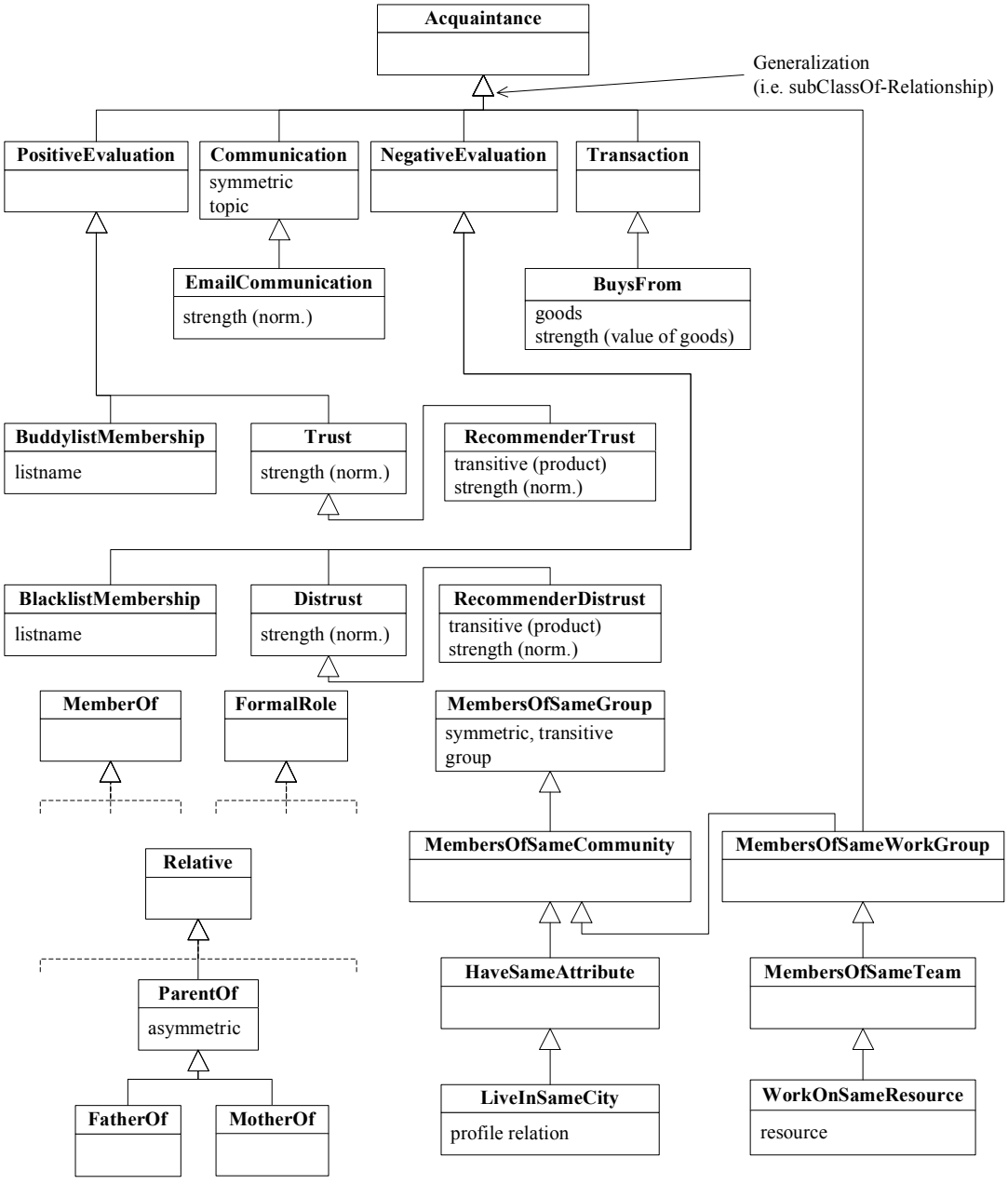


Figure 5.8: Hierarchy of relationship types

BuysFrom. Relationships of this type are special *Transaction* relationships. They posses an attribute "goods", referring to the goods transferred, and a strength value, which reflects the value of the goods.

MembersOfSameGroup. The relationship between two members of a group can be represented by the type *MembersOfSameGroup*. Relationships of this type are symmetric and transitive and may thus be used as equivalence relations for defining groups. The attribute "group" refers to the group the persons are members of. Note that this relationship type does not imply acquaintance relationships between the group members.

MembersOfSameCommunity. Compared to the previous type, this type adds the idea that two persons "share something (e.g. a language, network access, ...) but do not necessarily know each other or interact on a personal basis" (Schlichter et al., 1998). This type is a subtype of the previous type.

HaveSameAttributes. This relationship type is a subtype of *MembersOfSameCommunity*. It specifies the "shared thing" between members of the community — in this case equality of profile attributes. Subtypes of this type (e.g. *LiveInSameCity*) may define a profile relation explicitly defining the equality constraint.

MembersOfSameWorkGroup. This relationship type subsumes relationships among persons belonging to the same work group. According to Schlichter et al. (1998), belonging to the same work group implies knowing each other. The more general kind of group is a community (see above), the more specific one is a team (see below). This type is a subtype of *Acquaintance* and a subtype of *MembersOfSameCommunity*.

MembersOfSameTeam. According to Schlichter et al. (1998), "the members of a team know each other and are cooperating to achieve a common goal". The relationship between two members of a team can be represented by applying the type *MembersOfSameTeam*. This type is a subtype of *MembersOfSameWorkGroup*.

WorkOnSameResource. Being a subtype of the previous relationship type, *WorkOnSameResource* defines the additional attribute "resource" referring to the shared resource (e.g. jointly edited document).

MemberOf. This branch of types subsumes all kinds of membership relationships, i.e. someone's membership of a group, family, company, club, etc. According to the specific situation, specializations of this type may be useful, e.g. in order to indicate group leadership.

FormalRole. The type *FormalRole* represents relationships related to organizations, e.g. between leader and team, teacher and student, or doctor and patient. Formal roles are likely to be asymmetric (e.g. this is usually the case for superiority).

Relative. Kinship of any kind can be represented by a subtype of *Relative*, such as *ParentOf*, *FatherOf*, *SiblingOf*, or *MarriedTo*.

5.5.1 Remarks on the Type Hierarchy

The arrangement of relationship types in the hierarchy must be done with great care, in order to avoid contradictions to the common understanding of the types. Assume, for example, the type *AncestorOf*

is asymmetric and transitive, and *ParentOf* is a subtype of *AncestorOf*. Then, the transitivity property is also inherited by *ParentOf*. This is, however, a contradiction to the common understanding of parenthood.

To give a second example, assume that *MembersOfSameFamiliy* is a subtype of *MembersOfSameGroup*. Although *ParentOf* might be expected to be a subtype of *MembersOfSameFamily*, this may not hold true. Otherwise, *ParentOf* would inherit transitivity and symmetry from the supertype, which is, again, contradictory to the common understanding of parenthood.

5.5.2 Examples of Formalizations

The following example shows the formalization of the three relationship types *Acquaintance, Communication,* and *EmailCommunication*.

Acquaintance

```
<owl:Class rdf:ID="Acquaintance">
    <rdfs:subClassOf rdf:resource="#Relationship" />
    <rdfs:label xml:lang="en">Acquaintance</rdfs:label>
    <rdfs:comment xml:lang="en">
        This general relationship type represents the fact that
        a person knows the other. It does neither imply any
        evaluations of the other person, nor a face-to-face contact
        between the two persons.
    </rdfs:comment>
</owl:Class>
```

Communication

The restriction expresses that a communication relationship must have at least one topic keyword. Topic keywords may, for example, be extracted from message contents (emails, chat, IM, newsgroup postings) by applying standard keyword extraction procedures (e.g. Turney, 2003, 1997).

```
<owl:Class rdf:ID="Communication">
    <rdfs:subClassOf rdf:resource="#Acquaintance" />
    <rdfs:subClassOf rdf:resource="#SymmetricRelationship" />
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasTopic" />
            <owl:minCardinality
                rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="en">Communication</rdfs:label>
    <rdfs:comment xml:lang="en">
        Relationships of this type represent all kinds of communication.
        Communication relationships are assumed to be symmetric.
        The property hasTopic indicates the topics of the relationship
        as a set of string literals. Apply the property separately for
        each keyword. Supply at least one keyword for each relationship.
    </rdfs:comment>
</owl:Class>
```

```
<owl:DatatypeProperty rdf:ID="hasTopic">
    <rdfs:domain rdf:resource="#Communication" />
    <rdfs:range  rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

EmailCommunication

The additional restrictions define that an email-communication relationship has a strength value, which is normalized, and explicitly given by the strength function defined in the comment.

```
<owl:Class rdf:ID="EmailCommunication">
    <rdfs:subClassOf rdf:resource="#Communication" />
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#strength" />
            <owl:cardinality
                rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#normalizedStrength" />
            <owl:hasValue rdf:datatype="&xsd;boolean">true</owl:hasValue>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasStrengthSource" />
            <owl:hasValue rdf:resource="#Explicit" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="en">Symmetric email-communication</rdfs:label>
    <rdfs:comment xml:lang="en">
        This type represents special communication relationships by
        specifying email as the medium of the communication.
        Relationships of this type possess a normalized strength
        value as defined by the following formula:
        Let n12(k) be the number of emails sent (via to, cc, or bcc)
        in period k from person 1 to person 2, and n21(k) the number of
        emails sent in period k from 2 to 1.
        Period 1 is the current day, period 2 is the day
        before, and so on. Let S1(k) and S2(k) be the number of
        messages sent to any person by 1 and 2, respectively, in
        period k, and R1(k), R2(k) the number of messages received from
        any person in period k by 1 and 2, respectively.
        Let w(k)=(366-k)/(133590). Then, the strength s is defined as
        1/2 * sum(k=1,...,365) w(k) * ( n12(k)/S1(k) + n21/R1(k)
        + n21(k)/S2(k) + n12/R2(k) ). Periods in which any
        of the denominators is 0 or for which no data is available
        must be skipped in the sum.
    </rdfs:comment>
</owl:Class>
```

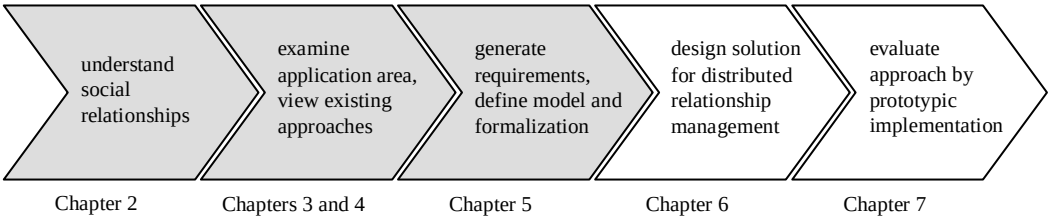


Figure 5.9: Roadmap of this thesis and intermediary results

5.6 Conclusions and Intermediary Results

This chapter has discussed requirements for social relationship management on the internet. On the basis of these requirements, a model for social relationships has been developed and formalized in terms of the Web Ontology Language OWL. Extending the semantics of the OWL ontology, semantics of algebraic properties have been defined in terms of a transformation algorithm for relationship knowledge bases. The ontology vocabulary has been applied for sketching the definition of basic relationship types.

What has been achieved so far in this work? Social relationships have been discussed, the application area has been explored, and a formal model for social relationships has been developed (cf. Fig. 5.9). The intermediary result up to here is *an ontology vocabulary for relationship information.* On the basis of the formal representation, it is possible for applications to exchange knowledge about those aspects of social relationships, which are relevant with respect to internet-based communication and shared information spaces. It has, however, not yet been discussed *how* the exchange of relationship information should take place — "how" refers to *social protocols* between entities storing social relationship information on behalf of one or several users.

Social relationship information is of great value — economists often refer to it as "social capital." The social protocols for the exchange of social relationship information must thus address privacy protection. This is one of the goals of the next chapter.

The following chapter addresses relationship management. The discussion of the application area in Chapters 3 and 4 has shown that relationship information originates from heterogeneous sources. Chapter 6 introduces mechanisms for integrating such relationship information. User agents are developed which manage relationship information on behalf of the users. These user agents for relationship management act on the basis of social protocols enforcing the compliance with privacy preferences of users.

Chapter 6

Distributed Relationship Management

For accomplishing everyday work we interact with a variety of communication and information systems. The social network between persons is not restricted to one specific context and one system, but distributed among several contexts and systems. Recognizing this distribution, in this chapter a strategy for interoperable relationship management is presented, which is based on a multiagent system. Special attention is paid to privacy protection. Two application examples show how the concepts developed in this chapter can be applied in order to improve reputation systems (cf. Chapt. 1) and to structure the own social network.

6.1 Distributed Relationship Information

Due to the variety of communication and information systems that we use for accomplishing everyday work, data about personal relationships is distributed among several applications. Summarizing earlier sections of this work (Chapts. 3 and 4, Tab. 3.2), some of these are:

- Email clients
- Usenet news clients
- Instant messaging and chat clients
- Semantic annotations of websites and documents (e.g. FOAF-powered home pages)
- Personal information managing (PIM) software (calendars, personal address books)
- Web platforms (e-commerce platforms, virtual community platforms)
- Groupware applications (group editors, shared work spaces)

For evaluation, exploration, or visualization of the personal social network, all of these applications must be taken into account.

Consider, for example, the problem of whom to trust on an internet-based auction platform. If a person a is interested in buying an item from person b, but does not know that person, a might use the relationship visualization component of the platform in order to find a relationship chain from a to b. Taking into account only "internal" relationships — i.e. those based on the interaction of users on the auction platform — leads to disregarding large parts of the social network among those users. This approach will thus probably not yield an optimal chain. If the relationship management component of the platform could also access relationship information concerning different contexts and originating

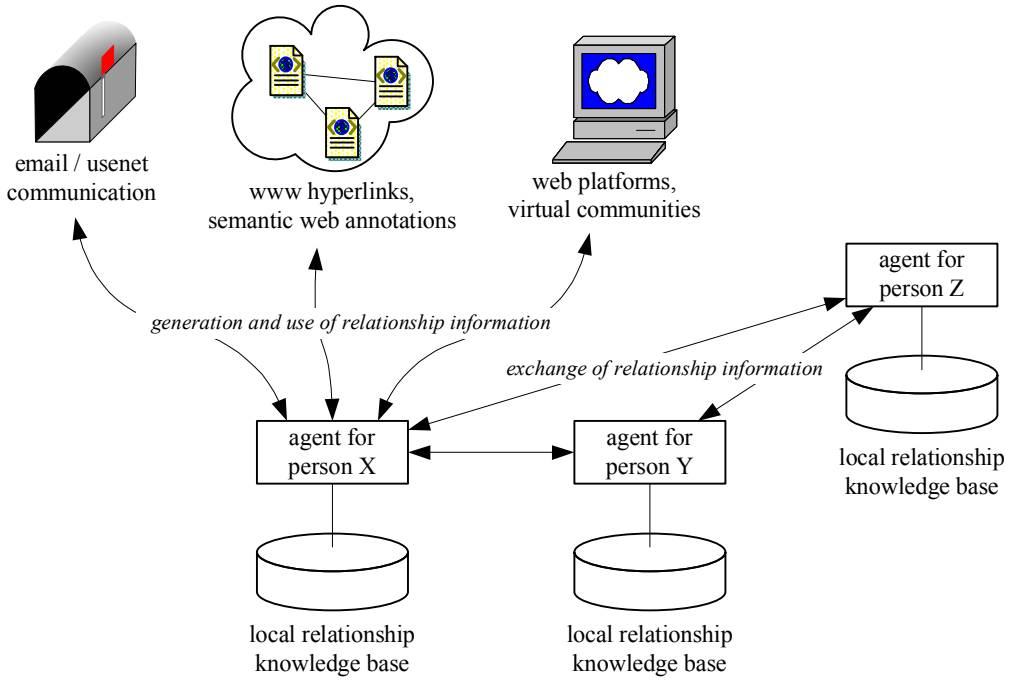


Figure 6.1: Sources of relationship information and exchange of relationship information among several agents

from different systems, the platform's model of the social network among its users would be much closer to reality and thus yield more optimal chains.

Yet, the direct exchange of relationship information between applications is not possible in practice. Every application would have to exchange relationship information with all other applications used by its users. Although this may be feasible in a "local" network of a few central applications (e.g. inside a company), it is certainly not in a large scale approach. The list of applications used by a person is not available to the applications due to privacy protection. Furthermore, many applications are run on local computers behind firewalls (e.g. email clients, PIM, etc.), and cannot be accessed by remote applications.

This work thus suggests an *agent-based* approach (see Sect. 6.6 for a discussion of software agents and agent-based relationship management), where all relationship information concerning a person is managed by a personal relationship management agent. Applications generating or using relationship information consult these agents via their *relationship management extension* (relationship management extensions are discussed in Sect. 6.4, see also Fig. 6.3). Agents communicate with each other and exchange relationship information in order to explore the social network, or to find relationship chains from one person to another. Figure 6.1 depicts this approach. This work does not assume a 1:1 but a 1:n mapping between agents and users. Especially in those cases, where users' computing devices are shielded by firewalls or not permanently "online" — as is the case for most private and office PCs — there may be agencies, run by third parties, which manage relationship information on behalf of several users.

Agents apply the relationship ontology vocabulary presented in Chapt. 5 for exchanging relationship information. Each application generating or using relationship information is equipped with an

extension for transferring relationship information to the agent and retrieving relationship information from the agent. Agents may thus be viewed as mediators for the exchange of relationship information between applications.

Apart from the formalization of relationships, important features of this approach are:

- Identity management — mapping different identities of the same user
- Design of the relationship management extension linking applications and agents
- Privacy — shielding relationship information from unauthorized access

The rest of this Chapter is organized as follows. In Sect. 6.2, basic mechanisms for user profile management are discussed. User profiles are referred to in relationship types declaring a profile relation — relationship management applications using such types must thus access user profiles.

Section 6.3 deals with the problem of linking a person's local user IDs which are used for accessing applications with the global user ID used by the relationship management system. A high-level protocol is presented which establishes the identity federation between the global ID and the various local IDs of a user.

Additional high-level protocols for the exchange of relationship information between applications and relationship management agents are discussed in Sect. 6.4.

Section 6.5 addresses the importance of privacy protection in the field of relationship management. A relationship-centric and rule-based approach to access control for relationship information is presented.

Building upon the preceding sections, Sect. 6.6 presents a multiagent system for relationship management. High-level social protocols for the exchange of relationship information between relationship management agents are defined.

In Sect. 6.7, solutions for the two scenarios "assessing the reputation of other users" and "structuring the own social network" presented in Sect. 1.6 are suggested, which is based on the multiagent system. Section 6.8 compares the approach of this thesis with existing work.

6.2 Storing and Accessing User Profiles

Section 5.1.4 has introduced profile relations for declaratively defining the conditions for the existence of a relationship between two persons. In the definition of a profile relation, user attributes are referenced. In order to evaluate of the profile relation, applications need access to the values of those attributes. They must therefore contact the entity storing the user profile.

Concerning the storage of and access to user profiles, this thesis adopts the approach of Koch and Wörndl (2001). According to their work, user profiles are stored in *ID-Repositories*.[1] The user retains full control over his profile — i.e. he defines who may access which parts of the profile and is aware which services have been given which parts of the profile. ID-Repositories are arranged in an identity management network (see Fig. 6.2) and may be operated by different organizations. This allows the user to choose an organization he trusts for storing his profile data. User profiles may, however, be cached by other repositories (taking into account the profile owner's privacy preferences) in order to improve the scalability of the approach.

[1] In order to differentiate the novelties of this work, this thesis assumes a conceptual (and possibly physical) separation of relationship management components and user profile management components. It is, however, likely that in the future, there will be components for both user profile management and relationship management.

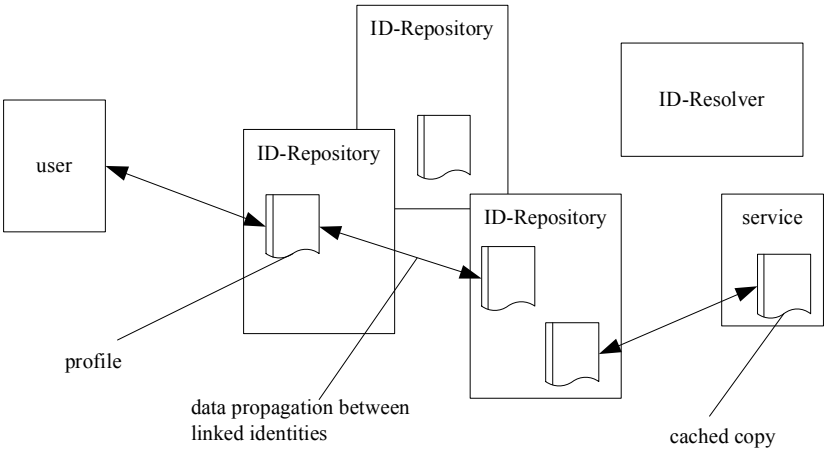


Figure 6.2: An identity management network, adopted from Koch and Wörndl (2001)

Each user profile is associated with one or more user IDs. In the case of this work these user IDs are URIs as described in Sect. 5.1.2. The mapping of user URIs to the associated profiles is maintained by the *ID-Resolver* service. For scalability, the ID-Resolver service may be replicated.

The vision of global identity management assumes that all user-related information is stored in the user profile — also including relationship information — and accessible via standards for user profile management. This is, however, not reality today. One reason is the lack of an accepted standardized representation of user attributes and their values. Yet, there are promising approaches like the Platform for Privacy Preferences (P3P), which includes a standardized representation of user profiles (Cranor et al., 2002). In the context of the semantic web, one of the main drawbacks of P3P is the lack of a semantic model for profile attributes. Most attributes (e.g. the user's address, or his employer) are represented as unstructured values. It is thus difficult to compare these attribute values automatically. The problem of generic user modeling is, however, a complicated issue on its own and cannot be discussed in detail in this thesis (see, for example, Kobsa, 2001). The following paragraphs therefore just present simple examples which have to be adopted and extended to implementations of future standardized user profiles.

With respect to relationship management, the following kinds of user profile data are especially interesting:

- Private and business addresses
- Interests
- Skills/profession

Private and business addresses. Address data is usually represented by a set of strings containing street, number, zip code, city, state or province, and country. Additionally, geo-coordinates may be provided.[2] Common examples of relationships based on address data are living in the same street, city, or region, or working in adjacent offices.

[2]Geo-coordinates and related issues are, for example, addressed in the OpenGIS Geography Markup Language (Open GIS Consortium, Inc., 2002). Details of encoding geo-references and coordinate transformations are not discussed in this thesis.

Interests, skills and profession. Interests and skills are typically set-valued attributes. These sets may either contain linguistically pre-processed keywords or classes or individuals of an ontology. Examples of relationships based on this type of profile data are sharing interests or skills, or pursuing the same profession.

Other profile data which can be relevant for relationship management are affiliations with companies or organizations (these can be straightforwardly transformed into relationships), PIM (personal information manager) information (e.g. calendars, address books, or special buddy-lists), transaction histories, and ratings. Without a standardized representation of such data it is, however, of little use to discuss these kinds of data in more detail.

Once the user profile scheme is defined, each user attribute is accessible via its identifier (a URI in this work). Suitable URIs are LDAP URIs (especially in those cases where the user ID is an LDAP URI, too), http URIs, or specific URI schemes depending on the profile scheme (e.g. a URI scheme for the P3P profile scheme might be defined).

Note that for any attribute whose values are classes or individuals of an ontology, the ontology must be known to the application. There are two possibilities to achieve this:

- Each ID-Repository provides a list of ontologies used and URLs where to retrieve them.
- URLs for ontologies are registered at a central ontology registry.

Locating the ontology for a given URI is, however, not a problem of social relationship management alone, but of the whole semantic web. Future work is expected to address this issue. For this work, it can be assumed that each ID-Repository provides a registry of URLs of relevant ontologies.

6.3 Identity Federation

For different applications, a person may have different user IDs — e.g. an email-address for email- and usenet-accounts, a special email-address used for a Microsoft Passport[3] account, a URI used for semantic web annotations, or several local user IDs for community platforms or groupware applications.

Currently, standards are being developed which aim at linking these different user IDs in order to increase interoperability (e.g. Microsoft Passport, Liberty Alliance[4], live ID[5], identity management components in Sun ONE[6], or Extensible Name Service[7]). Yet, since none of these approaches have been widely accepted so far, it is not likely that in the near future all user IDs of a person will be linked.

Hence, the multiagent system for relationship management presented in this chapter must also link the different user IDs of a person. For each person, there needs to be a user ID (the *global ID*) for the relationship management multiagent system which is used for the exchange of relationship information between different agents and between an agent and an application. In practice, it is convenient if this is the same user ID as that used for accessing the user's user profile (cf. Sect. 6.2). The relationship management extension (cf. Sect. 6.4) of each application generating or using relationship information maintains a mapping of global IDs to the corresponding local IDs (cf. Fig. 6.3).

[3] http://www.passport.com
[4] http://www.projectliberty.org
[5] http://www.live-id.org
[6] http://www.sun.com/software/sunone
[7] http://www.xns.org

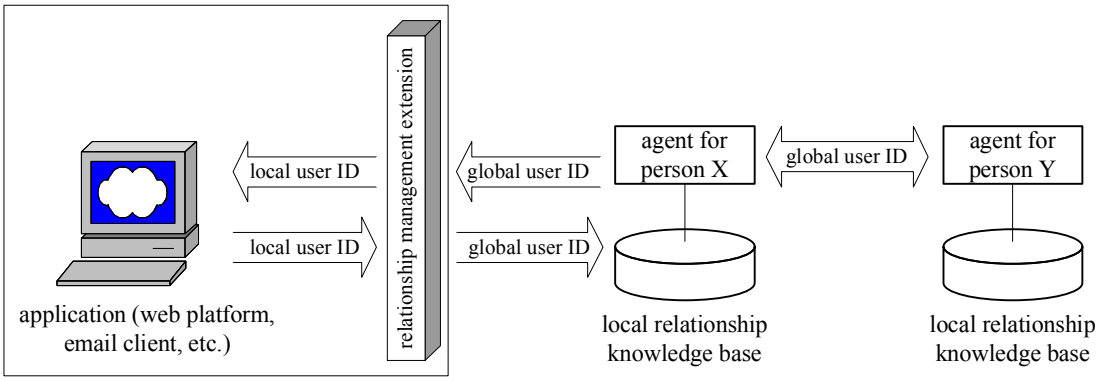


Figure 6.3: Global and local user ID

On a conceptual level, the federation of a local ID and the corresponding global ID is established as follows.

1. A user logs into an application supporting relationship management.
2. The user is asked if he wants to federate his local ID with his global ID for relationship management.
3. If the user chooses to do so, he enters his global user ID and a reference to his user agent. The reference to the agent may not be necessary if a global registry for user IDs and agents is available — this depends, however, on the actual implementation of these concepts.
4. The relationship management extension (RM extension) of the application contacts the user agent via a secure connection. The RM extension and the agent exchange an authentication handshake (e.g. based on X.509 certificates). The RM extension sends the message `ID-federation-request(GlobalID gID, Date expirationDate)` for identity federation, including a requested expiration date.
5. The user agent contacts the user and asks him if he wants to accept the request, and how long the federation should be valid.
6. If the user accepts and enters an expiration date, the agent generates an *access ticket* — i.e. a representation of the access rights granted to the RM extension — and delivers it to the RM extension, accompanied by the expiration date, via the message `ID-federation-accepted(AccessTicket at1, Date expirationDate)`. Otherwise, the request is rejected by sending the message `ID-federation-rejected()`.
7. The RM extension generates an access ticket and sends it to the agent, via the message `ID-federation-confirmed(AccessTicket at2)`.
8. The local ID and the global ID are now federated up to the expiration date the user entered.

From now on, every time the remote application wants to access relationship information concerning the user, it contacts the corresponding agent via the RM extension, provides his access ticket, and may

then access the required information (if the access ticket has not yet expired). Vice versa, if an agent requires relationship information from the application, it contacts the application's RM extension, uses his access ticket, and may then access the information.

Relationship information that an application has retrieved from an agent may under no circumstances be delivered to other users. It may only be used for processing a task the owner of the relationship information has initiated. In the case of online auction houses, for example, the auction platform may contact the relationship agent in order to find relationship chains between the user and another person and present those chains to the user. The auction platform may not deliver this information to any other user. It may, however, store it in a local cache for some time in order to improve performance.

6.4 The Relationship Management Extension

Relationship management extensions of applications have been repeatedly mentioned, without explaining them in more detail. The extension links the application and the relationship management multiagent system — it may thus be viewed as a special agent. Its functions are the following:

- Establish ID federation
- Revoke ID federation
- Deliver relationship information stored locally to an agent
- Retrieve relationship information needed by the application from agents
- Integrate ontologies for relationship management

In the remainder of this section, these functions are described on a conceptual level. Instead of an actual implementation, messages are described that are exchanged between the extension and an agent. Implementations of these concepts depend on the agent framework and the application and may, for example, apply RPC implementations, Enterprise Java Beans (EJB)[8], or SOAP[9].

Establish an ID federation. Upon initiation by the application, the relationship management extension contacts the user's agent in order to establish the federation of the local ID and the global ID, as described above.

Revoke ID federation. The federation of a local and a global ID may be revoked by both parties at any time — e.g. because a user deregisters at a web platform, or the user does not want the two identities to be federated any more. The message flow for federation revocation is:

1. The initiator of the revocation (agent or application) sends the message `revoke-ID-federation(GlobalID gID)`.
2. The other party replies with `revocation-confirmed()`.

The revocation of an ID federation does not affect any relationship information exchanged before.

[8] http://java.sun.com/products/ejb/
[9] http://www.w3.org/TR/SOAP/

Deliver relationship information to agents. There are two strategies for transferring relationship information from applications to agents: an extension-triggered approach and an agent-triggered approach.

Extension-triggered approach. This approach assumes that the extension informs the agent about new relationship information. Messages are exchanged as follows.

1. The extension sends `update-RI(GlobalID gID, RDFdata update, RDFdata remove)`.
2. The agent processes the update message (i.e. it adds/removes all relation information contained in the RDF models represented by `update` and `remove`, respectively) and replies `update-confirmed()` on success. If any problems occur (e.g. any resources are currently not accessible), it replies `update-failed()`. The extension must then try to send an update-message later on. The list `update` contains relationship information which has been changed or newly created since the last update. The second list `remove` contains relationship information that has been invalidated since the last update.

Agent-triggered approach. According to this approach, an agent decides when it needs updated relationship information from the application. The message flow of this approach is the following.

1. The agents sends `request-update-RI(GlobalID gID, Date startdate)` to the extension of the RI application.
2. The extension replies `update-RI(GlobalID gID, RDFdata update, RDFdata remove)`, including all relationship information that has been added, changed, or invalidated since `startdate`.
3. The agent processes the update message and replies `update-confirmed()` on success. If any problems occur, it replies `update-failed()`, and may try again later on.

The `RDFdata`-structures used in the messages above contain valid RDF-serializations of the corresponding relationship information — e.g. in RDF/XML. Which serialization syntaxes are valid must be defined by an actual implementation of these concepts.

An application does not have to support both approaches. An instant messaging application offering relationship information about buddylist membership, for example, may decide to support only the extension-triggered approach, because updates of relationship information are rather rare. On the other hand, a community web platform should support the agent-triggered approach, because agents running on a personal computer behind a firewall may not be accessible from outside the local area network they belong to.

Retrieve relationship information from agents. The RM extension of an application may contact an agent in order to retrieve relationship information. The four possible kinds of queries the extension may send to an agent are the following:

- Return a list of relationships from person x to a person y:
  `query-RI-to-person(GlobalID x, GlobalID y)`

- Starting from person x, explore the social network up to depth d, taking into account relationship information of a certain kind, only: `query-explore-network(GlobalID x, RITemplate template, Integer d)`[10]
- Find relationship chains of a certain kind from person x to person y with maximum length d — i.e. return all relationships the chains consist of: `query-chains-to-person(GlobalID x, GlobalID y, RITemplate template, Integer d)`
- Check, if a relationship chain of a certain kind of relationship with maximum length d exists from person x to person y — i.e. do not return all relationships the chain consists of: `query-exists-chain-to-person(GlobalID x, GlobalID y, GlobalID initiator, RITemplate template, Integer d)`

The messages exchanged for these queries and the structures therein are the same as those used between two relationship management agents of different persons. Hence, the description of the messages is delayed until Sect. 6.6.3.

Integrate ontologies for relationship management. The RM extension of an application is responsible for translating relationship information from the application-internal representation to the formal representation and vice versa. It must thus contain reasoning components for relationship ontologies, algorithms for resolving URI references, and methods for retrieving the associated ontologies from the web.

6.5 Privacy

Relationship information is personal data of high value. We use our relationship network for accomplishing everyday work — we seek expertise and advice from people we know, and establish new relationships with the help of people we know. In a professional setting, the relationship network of managers makes up a huge amount of "social capital", and plays an important role for success or failure.

Recognizing the high value of relationship information, it is obvious that a system facilitating the management and the exchange of relationship information must consider it's users' privacy. Those aspects of privacy concerning personal data are subsumed under the term *information privacy*. According to Clarke (1999),

> information privacy refers to the claims of individuals that data about themselves should generally not be available to other individuals and organizations, and that, where data is possessed by another party, the individual must be able to exercise a substantial degree of control over that data and its use.

Other aspects of privacy — e.g. concerning the physical person and the "right to be let alone" (Warren and Brandeis, 1890) — are of minor importance with respect to this work. Hence, in the following, the term "privacy" always refers to information privacy.

According to Clarke's definition, privacy encompasses access control and a substantial degree of control over the use of personal data such that personal information cannot be accessed by third parties which have not been authorized for the use of that data. The following subsection thus gives a quick introduction to the basics of access control. Afterwards, a rule-based approach to privacy management in the field of relationship management is presented.

[10] The structure `RITemplates` refers to relationship information templates as described in Sect. 5.3.8.

	o_1	o_2	o_3	o_4
s_1	read		read, write	read
s_2		read, write	read	

Table 6.1: An access control matrix for two subjects and four objects

6.5.1 Basic Principles of Access Control

Access control refers to means for dealing with requests of subjects to access certain objects within a system / application. The following three classes of entities can be distinguished:

- Subjects: (active) entities which access objects
- Objects: (passive) entities which are accessed by subjects
- Actions: different kinds of access to objects (e.g. read, write, create)

With respect to relationship management, subjects correspond to users (resp. their agents), and objects correspond to individual pieces of relationship information.

The core artifact for access control is the access control matrix (ACM). Each row of this matrix corresponds to one subject, whereas each column corresponds to one object. The entries in the ACM define the allowed actions for each subject-object-pair. Table 6.1 shows an example of an ACM.

Implementations of the ACM may be either done row by row, or column by column. The former approach corresponds to linking an access control list (ACL) to each object, whereas the latter leads to issuing documents containing a representation of the access rights granted to the user with respect to each object (capabilities, “access tickets”). For details see, e.g., Coulouris et al. (2001).

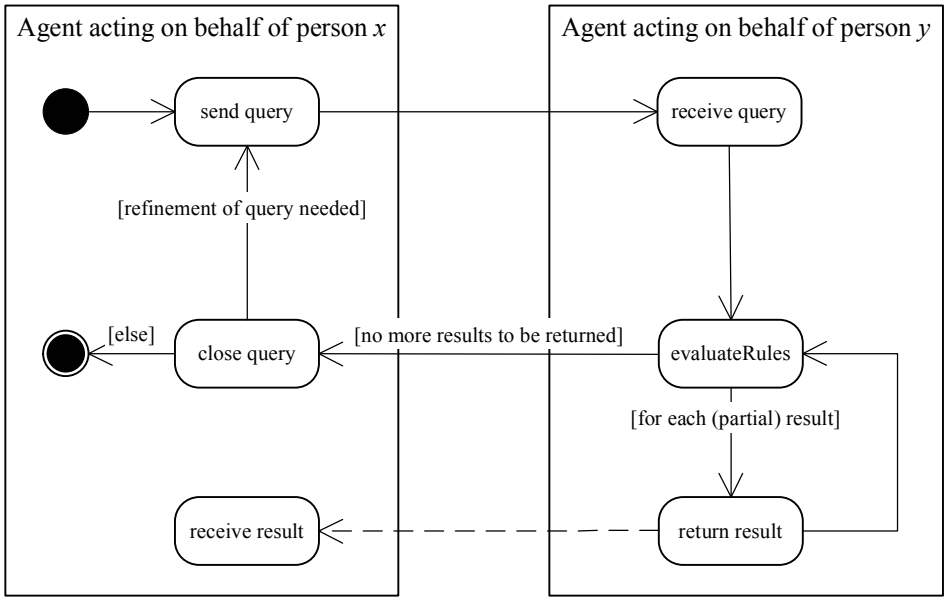
Both approaches have certain drawbacks. Access tickets are usually independent from the person they were issued to, in order to allow the delegation of access rights. Hence, if a ticket is stolen, the thief can use it in order to gain access to the object the ticket was issued for. Furthermore, it is difficult to revoke access tickets.

Access control lists, in contrast, are safe against ticket theft, but are far less flexible. This approach is thus often extended by assigning *roles* to subjects (role-based access control, RBAC). In this approach the rows of the ACM correspond to the roles, and allowed actions are assigned to role-object-pairs (see Ferraiolo and Kuhn, 1992, for details).

It is interesting that, in all of these approaches, access rights are either defined explicitly (ACLs) or via the introduction of additional artifacts (capabilities, RBAC). Relationship information is, however, highly dynamic information. It quickly changes, new relationships are established (this leads to additional columns in the ACM), relationships fade (drop the corresponding columns of the ACM), or formerly unknown persons move into the social network (add rows to the ACM). None of the approaches mentioned above can deal properly with these dynamics without the need of highly frequent user-interaction (explicitly defining access rights, assigning roles to persons, or issuing access tickets). The following subsection thus suggest a relationship-centric and rule-based approach to access control for relationship information.

6.5.2 Rule-based Privacy Management for Relationship Information

For defining access rules for relationship information, this work suggests a relationship-centric approach. Different from classical solutions, this approach is neither focused on the subjects accessing



Boxes denote states, arrows indicate state transitions. The dashed arrow indicates a flow of data without transferring control. The solid circle indicates the start state, the solid circle with an additional circle surrounding it marks the end state.

Figure 6.4: General state diagram illustrating the processing of an RI request between two agents

relationship information, nor on the objects (i.e. individual pieces of relationship information). Instead, it focuses on the *relationship* between the subject accessing relationship information, and the owner of that information. Considering the two aspects of privacy mentioned by Clarke, privacy management must provide answers to the following two questions:

- Who may access certain pieces of relationship information?
- What may an agent do with relationship information received from another agent?

In order to answer both questions, each relationship management agent possesses a set of rules for each user the agent is acting on behalf of, defining who may access what kind of relationship information and for what purpose. Based on these rules, a query of relationship information between two agents is processed as follows.

1. Agent 1, acting on behalf of person x, requests relationship information from agent 2, acting on behalf of person y.
2. By evaluating the set of access rules agent 2 decides which pieces of relationship information matching the request may be returned to agent 1.
3. Agent 2 returns those pieces of relationship information to agent 1, accompanied by a restriction on the allowed usage.

The overall processing of a request for relationship information is depicted in Fig. 6.4. In more detail, the access rules consist of the four components rule priority, domain, condition, and allowed usage:

Rule priority. The priority defines the order for processing the rules. 1 is the highest priority, larger numbers indicate lower priorities. Rule priorities define the order, in which the rules are applied. For each priority, there may thus be at most one rule.

Domain. The domain describes to which kinds of relationship information the rule applies. It is expressed as a relationship information template as defined in Sect. 5.3.8. The domain of an access rule for communication relationships, for example, might be defined by the RDF graph depicted in Fig. 6.5 a. An example of a more complicated domain of an access rule for specific group relationships is shown in Fig. 6.5 b.

Condition. The condition is a conjunction of n chain expressions each consisting of a relationship information template t_i, and an integer d_i. n may be 0 in order to indicate an empty condition, which is always fulfilled. If agent 1, acting on behalf of person x, requests relationship information from agent 2, acting on behalf of person y, agent 1 may access the requested relationship information, if for all $i, 1 \leq i \leq n$, there exists a relationship chain of maximum length d_i between x and y, where each individual relationship of that chain matches t_i (cf. Sect. 5.3.8). Note that it is possible to express disjunctions of chain expressions by defining a separate rule for each term of the disjunction. Figure 6.6 depicts an RDF graph of a rule whose condition consists of two chain expressions.

Allowed usage. The possible values for this element are *public*, *anonymous*, and *private*. Assume the condition of a rule r is fulfilled. Let R refer to those pieces of relationship information passed to the requesting agent in accordance with r. Then, the allowed usage element of the rule r defines, what the requesting agent may do with the relationship information:

- *public*: R may be used and passed to other agents and applications as desired by the requesting agent.
- *anonymous*: R may be used by the requesting agent, but only be passed to other agents and applications if the user ID of the owner of the relationship information is replaced by a random pseudonym. R may be used for forwarding a query concerning the existence of a relationship chain (cf. Sect. 6.6.3).
- *private*: R may only be used internally by the agent and must not be passed to other agents or applications, not even indirectly by forwarding a query concerning the existence of a relationship chain.

If a rule is applied and the relationship type of the rule domain is a supertype of the relationship type of the relationship information being accessed, only the information corresponding to the supertype may be returned — i.e. the relationship information must be casted to the supertype. If a trust relationship is accessed and there is no rule with the corresponding domain, for example, and the rule for acquaintance relationships is applied instead (assuming that acquaintance is a supertype of trust), the trust relationship must be dealt with as if it was an acquaintance relationship — i.e. in the result, the type must be refined and additional attributes and properties which are not defined by the acquaintance relationship must be removed.

It is important to distinguish between conditions of depth 1 (i.e. $d_i = 1$ for all terms of the conjunction) and conditions of higher depths. Conditions of depth 1 can be evaluated on the basis of the knowledge of the agent itself without the need to query other agents. Checking conditions of depth 1 is thus much more efficient than checking conditions of higher depths. As will be discussed in the following section, the checking of conditions of higher depths may not even always terminate. For simplifying language, rules are classified according to their conditions:

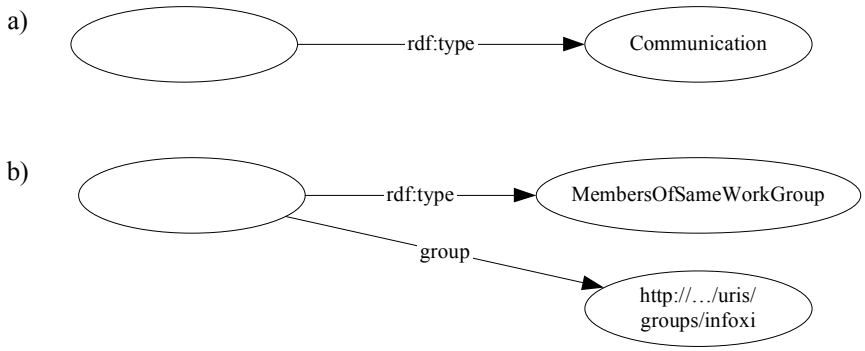


Figure 6.5: Two examples of domain definitions for access rules

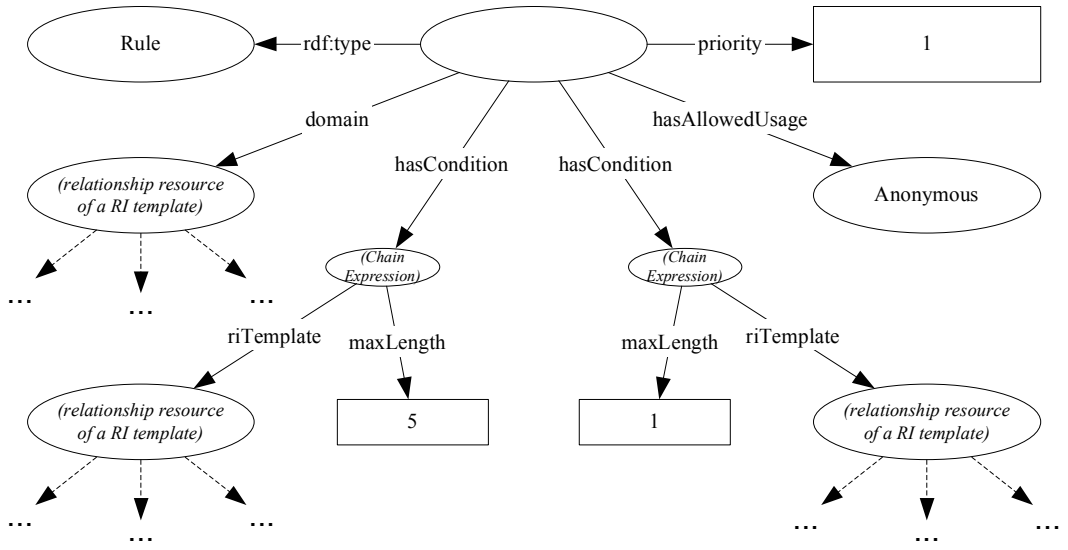


Figure 6.6: RDF representation of an access rule for relationship information possessing a condition consisting of two chain expressions (represented by the two resources of type `ChainExpression`) with maximum lengths 5 and 1

- A *level 0 rule* is a rule with an empty condition.
- A *level 1 rule* is a rule with a condition of depth 1.
- All other rules are *level 2 rules*.

Concerning the rule syntax, this work suggest to represent rules in terms of OWL/RDF, for the sake of uniformity with the rest of relationship-related data. Figure 6.6 depicts the RDF graph of a rule definition (see also Fig. 6.11, p. 122). The rule ontology is presented in App. B.

6.5.3 Relationship-centric Access Control in Other Applications

Relationship-centric access control is not limited to relationship management — it may be applied in other application domains as well. A simplified version — based on buddylist-relationships — has been applied in the Cobricks user management module (cf. p. 40 et sqq.). Based on the Cobricks relationship management module sketched in Chapt. 7, a Cobricks rule processor (cf. Fig. 3.7, p. 44) can be implemented allowing for the use of relationship information templates in rule conditions.

Integrating relationship management with a general identity management architecture, such as that suggested by Koch and Wörndl (2001), see also Fig. 6.2, is a promising approach. Access rules for user profile data might be defined in terms of relationship information templates, offering both a high flexibility and a high expressiveness.

Apart from these two examples, relationship-centric access control may also be applied to shared information spaces, offering the main benefit of gaining flexible and expressive access control without having to introduce additional artifacts like groups, or access tickets. This work will, however, not go into detail here.

6.6 A Multiagent System for Relationship Management

The term “agent” has been repeatedly used in this chapter without defining it properly. Figure 6.1 has suggested the existence of several agents acting on behalf of different persons and exchanging relationship information. This section elaborates on the multiagent system. At first, it throws a glance at the term “agent”, and then defines messages and (social) protocols between the agents for exchanging relationship information.

6.6.1 Software Agents

There has been a considerable amount of research on software agents (e.g. Weiss, 1999; Bradshaw, 1997; Foner, 1997, 1993). This thesis cannot even try to summarize the most important approaches discussed in that work. It can, however, briefly discuss common characteristics of software agents and how these attributes apply to relationship management agents.

In fact, there is still no agreement upon a definition of the term “software agent”. For this work, it is appropriate to view a software agent as a special piece of software exhibiting certain characteristics. Wooldridge (1999) emphasizes the following important characteristics of intelligent agents[11]:

Autonomy. An agent has control over its own internal state and its behavior and pursues a certain goal. In the case of relationship management, this goal is to manage the user’s relationship information, keep it up-to-date, gain new knowledge about the user’s social network, and protect the user’s privacy.

[11] For the scope of this work, the terms “software agent”, “intelligent agent”, and “agent” are synonyms.

Reactivity. An agent can react on changes of its environment — e.g. incoming messages from other agents or applications. In the case of relationship information, this characteristic refers to answering queries from other agents or applications.

Proactiveness. This characteristic expresses that an agent is capable of taking the initiative in order to pursue its goals — e.g. for updating relationship information or gaining additional knowledge about the user's relationships.

Social ability. Agents are usually expected to exhibit social behavior. Relationship management agents, for example, collaborate to check conditions of access rules, or to answer queries concerning the disclosure or existence of relationship chains.

These four characteristics are mostly agreed upon in standard literature about software agents. Additionally, the ability to learn is frequently mentioned in connection with intelligent agents (Weiss, 1999; Bradshaw, 1997; Borghoff and Schlichter, 2000) — in the case of this work, an agent learns by integrating relationship information gathered from applications or other agents into its own knowledge base.

There are several other characteristics software agents may exhibit, e.g. mobility, personality, adaptivity (Bradshaw, 1997). This thesis, however, will not go into detail here.

6.6.2 General Purpose and Design of the Multiagent System

The overall goal of the multiagent system is the management of relationship information. From the perspective of each individual agent, acting on behalf of one or several user(s), this means the following:

- Store relationship information concerning the users' social networks
- Keep that information up-to-date and detect new relationships
- Control access to and usage of relationship information
- Answer queries for relationship information from (federated) applications
- Cooperate with other agents in order to increase the own knowledge about the users' social networks

The environment the agents are situated in thus consists of all agents and applications supporting relationship management. The domain of discourse of the agents is relationship information in the formal representation presented in Chapt. 5.

Types of Agents and their Functions

In more detail, the following types of agents can be distinguished:

User agents. These are agents managing relationship information on behalf of one or several user(s). As has been mentioned above, it may be useful not to run agents on personal desktops or mobile devices, but on a trusted server, in order to increase the availability of the agent. The overall goals of agents of this type have been mentioned before.

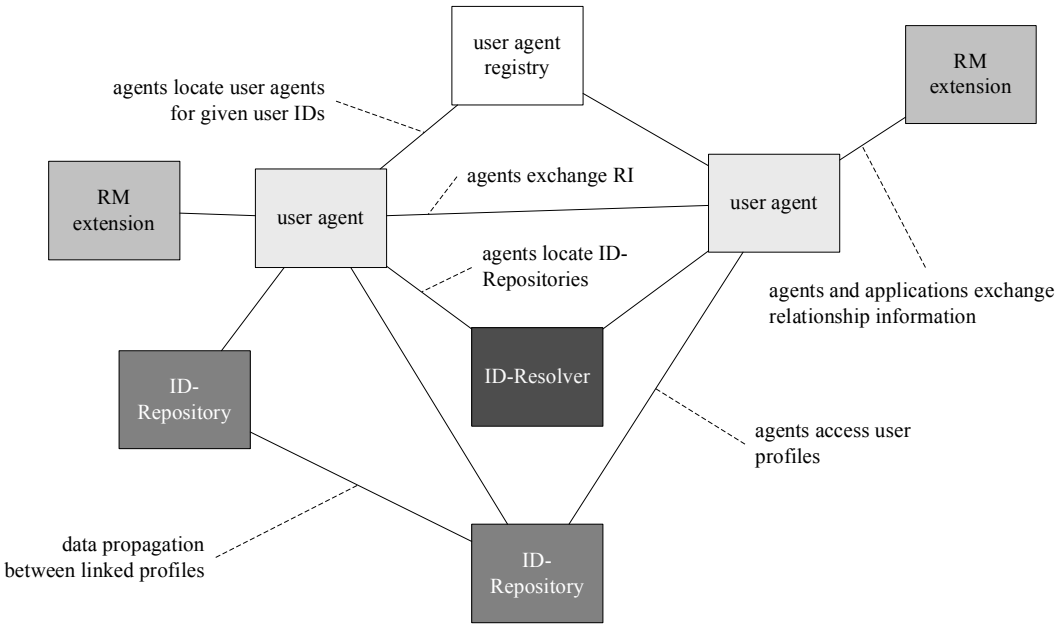


Figure 6.7: Types of agents in the relationship management multiagent system

RM extensions of applications. This type of agent fulfills the functions of a relationship management extension, as described in Sect. 6.4. It is somewhat questionable if a relationship management extension should be viewed as an agent, because its autonomy is rather limited. Yet, since RM extensions collaborate with user agents, it is convenient to view them as agents, too.

User agent registry. This agent is responsible for maintaining a registry of user agents and their associated user IDs. Other agents consult this agent in order to find the user agent acting on behalf of a certain person. The user agent registry may be replicated in order to enhance performance.

ID-Repositories. ID-Repositories are responsible for storing user profile data and providing access to user profile data to authorized users and services. With respect to relationship management, ID-Repositories are needed for the in order to evaluate of profile relations (cf. Sect. 5.1.4).

ID-Resolver. The function of this agent is to resolve the ID-Repository agent storing the profile of a certain user (cf. Fig. 6.2). Like the user agent registry, this service may be replicated in order to enhance performance. It is likely that in implementations, there will be agents fulfilling both the functions of the user agent registry and the ID-Resolver.

The following sections will elaborate on relationship management user agents. The RM extension has been discussed in Sect. 6.4. Main functions of ID-Repositories and ID-Resolvers have been sketched in Sect. 5.1.4 — the reader is referred to Koch and Wörndl (2001) and Koch (2003b) for details.

Implementation

There are a variety of approaches for implementing multiagent systems (see Mangina, 2002, for a review of relevant software prodcuts) — e.g. the standards of the Foundation for Intelligent Phys-

ical Agents (FIPA)[12] and the corresponding open source implementation FIPA-OS[13], or the JADE Framework[14] (also based on FIPA standards). The main concern of most of these approaches is the definition of a framework and a technical environment the software agents are situated in — e.g. in the case of FIPA, a message transport system, an agent management system (registry, "white pages"), and a directory service ("yellow pages"). Agents interact via the exchange of messages, whose basic structure is also defined in the approaches mentioned above. The contents of the message are, however, interpreted by the agents themselves and are thus dependent on the application scenario.

For the rest of this chapter, the agent framework is taken for granted — including a directory agent fulfilling the functions of the user agent registry. The following subsections thus concentrate on the contents of the messages exchanged between user agents.

6.6.3 Overall Behavior and Communication of User Agents

The general behavior of a user agent can be split up into reactive behavior and proactive behavior. In both cases, user agents may interact with other user agents or RM extensions.

Reactive Behavior

The reactive behavior defines the way an agent reacts on requests for relationship information from other agents. It has already been described in Sect. 6.5.2 (cf. Fig. 6.4).

Proactive Behavior

The proactive behavior of an agent refers to keeping the relationship information knowledge base up-to-date:

- If a piece of relationship information has expired, the agent contacts the application or agent the piece of relationship information originates from and asks for an update.
- The agent periodically requests new relationship information from other agents and applications.

Actual strategies of an agent depend on its purpose. Agents may pursue minimal strategies aiming at keeping the relationship knowledge base small, or exhaustive strategies aiming at gathering as much information as possible. The level of proactiveness of an agent is defined by its implementation and is a decision of the agent designer.

Communication between user agent and RM extension

The interaction between an agent and an RM extension has already been discussed in Sect. 6.4. The set of messages for retrieving relationship information from an agent, which had been skipped there, will be subject of the following subsection.

[12] http://www.fipa.org
[13] http://sourceforge.net/projects/fipa-os/
[14] http://sharon.cselt.it/projects/jade/

Communication between two user agents

Communication between two user agents is always initiated by an authentication handshake. Since a user agent may be acting on behalf of several users, it is necessary to state on whose behalf the agent is acting with respect to actual queries, and to whom (i.e. to which *person*) the following queries are addressed. The result of the authentication handshake is an abstract *communication channel* between the two agents. All communication across that channel is associated with the persons on whose behalf the authentication handshake has been exchanged. For the general behavior of user agents the technical details of the authentication handshake and the communication channel are, however, of minor importance, and are thus left to implementations of these concepts. From now on it is assumed that all query-related communication takes place via such abstract communication channels and is thus associated with two *persons*, not their agents only. An agent receiving a query may therefore only take into account relationship information it is managing on behalf of the respective person.

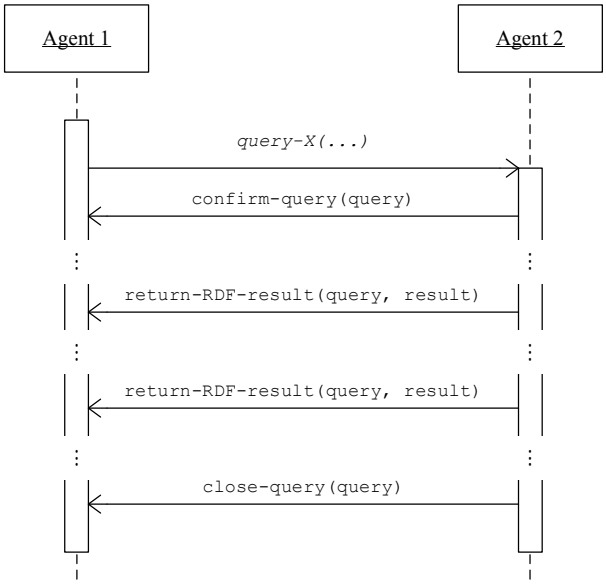
Each individual query is initiated by a query message, and (if the query terminates at all) terminated by a message indicating the closure of that query. Upon sending a query, the agent receiving the query confirms the query, generates a query ID and sends the query ID back to the requesting agent. The query ID is used in later messages referring to the query, which submit (maybe partial) results to the requesting agent, and in the closing message. Note that, for answering a query, an agent may submit several result messages to the requesting agent. As a basic principle, an agent should return partial results to the requesting agent, as soon as it has evaluated the corresponding access rules and decided that some pieces of relationship information may be submitted to the agents. The message flow between two user agents is depicted in Fig. 6.8. In more detail, the following messages are used in the communication between two user agents.

`query-RI-to-person(GlobalID x, GlobalID y)`
: This is a request for all relationship information between person `x` and person `y`. The results are RDF documents containing the RDF models of all relationships the agent wants to submit to the requesting agent. For submitting results, a `return-RDF-result`-message is used.

`query-explore-network(GlobalID x, RITemplate template, Integer d)`
: The result of this query is an RDF document describing the social network of `x` up to a depth `d`, where only relationships matching the RI template `template` are taken into account. For answering this query, the user agent may decide to initiate subqueries of the form `query-explore-network(y, template, d-i)`, in order to extend chains of length `i`, which have been found in the local relationship information knowledge base. Results (i.e. an RDF model of the network) are submitted via a `return-RDF-result`-message.

`query-chains-to-person(GlobalID x, GlobalID y, RITemplate template, Integer d)`
: According to this query, the agent searches for chains of maximum length `d` from person `x` to person `y`, where each relationship contained in the chain matches the RI template `template`. In order to answer this query, the user agent may decide to initiate subqueries of the form `query-chains-to-person(z, y, template, d-i)`, in order to extend chains of length `i`, which have been found in the local relationship information knowledge base. Results (i.e. an RDF model describing all chains which have been found) are submitted via a `return-RDF-result`-message.

`query-exists-chain-to-person(GlobalID x, GlobalID y, GlobalID initiator, RITemplate template, Integer d, Integer l)`



The message *query-X(...)* stands for one of the following messages (with appropriate parameters): `query-RI-to-person`, `query-explore-network`, `query-chains-to-person`, or `query-exists-chain-to-person`. In the latter case, agent 2 (or another agent the query has been forwarded to) replies with `return-chain-found` instead of `return-RDF-result`. Note that the evaluation of access rules may incorporate sending queries to other agents. Such queries are not shown in this figure.

Figure 6.8: Message flow following a query from agent 1 to agent 2

Like queries of the previous type, queries of this type aim at finding a chain of maximum length `d` from person `x` to person `y`, where each relationship contained in the chain matches the RI template `template`. The parameter `l` indicates the length of the chain. The agent may initiate subqueries of the form `query-exists-chain-to-person(x, y, initiator, template, d-i, l+i)`, in order to extend chains of length `i`, which have been found in the local relationship information knowledge base (the initial `l`-value for new queries is 0). An important difference to the previous queries is that results are not sent to the requesting agent, but to the user agent acting on behalf of the person referred to as `initiator`. The parameter `initiator` may be omitted (i.e. set to `null`) — in this case results are sent to the requesting agent. For submitting the result, a `return-chain-found`-message (with `length` $=$ `l`) is used. If no chain can be found, the closing message is sent without having sent a `return-chain-found`-message before.

`confirm-query(QueryID query)`
: By sending this message, an agent confirms the acceptance of a query sent to him. The parameter `query` is a query ID, which will be used in later messages concerning the query.

`return-RDF-result(QueryID query, RDFdata result)`
: This message is used for returning results of a query. The parameter `query` is the query ID of the original query, and the parameter `result` is an RDF graph containing the (maybe partial) result of the query.

`return-chain-found(QueryID query, Boolean chainFound, Integer length)`
: Queries for the pure existence of a chain are answered by this message if a chain is found. The function of the parameter `query` is the same as in the previous message. The integer `length` indicates the length of the chain.

`close-query(QueryID query)`
: This message closes the query with the query ID `query` — i.e. no further result messages will follow.

6.6.4 Agent-internal Processing of Requests and Resolution of Access Rules

The internal components of a user agent are the following (cf. Fig. 6.9).

Communicator. This component is responsible for the communication with other agents in the multiagent system. The messages used for communication have already been discussed.

Agent control. The agent control component defines the behavior of a user agent. The internal control algorithms for answering queries and evaluating access rules are sketched below.

List of queries to answer. This list contains all queries that have been received and confirmed by the agent, but not yet closed. A query is stored on this list, if the agent decides to initiate subqueries for answering the query and is waiting for the results of these subqueries.

List of processed queries. Queries for the existence of a relationship chain, which have been processed by an agent are stored on this list for some time.

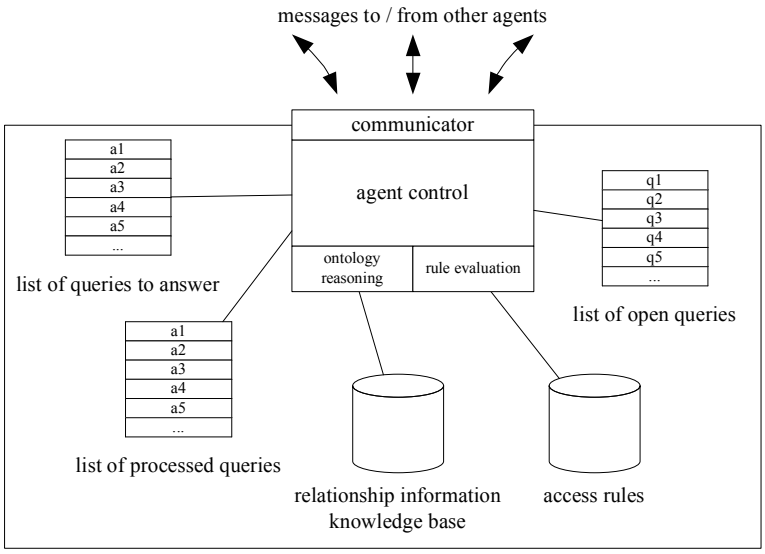


Figure 6.9: Internal structure of a user agent

List of open queries. Every time an agent sends a query to another agent and receives the corresponding confirmation, the query is stored on this list. Upon receiving a closing message, the corresponding query is removed from the list.

Relationship information knowledge base. This is the set of relationship information and relationship ontologies managed/known by the agent. The ontology reasoning component is used for querying the RI knowledge base.

Access rules. Every agent stores a set of rules defining who may access which kinds of relationship information. Access rules for relationship information have already been discussed in Sect. 6.5.2. They are defined by the user the agent is acting on behalf of via the agent's user interface.

In the remainder of this subsection, the agent control component is discussed in more detail.

Processing of a Query of Relationship Information

A query q of an agent acting on behalf of person k, where q is one of the queries `query-RI-to-person`, `query-explore-network`, or `query-chains-to-person`, is processed as follows (cf. Figs. 6.4 and 6.10).

1. The agent retrieves relevant relationship information from its RI knowledge base.

2. For each piece of relationship information, the agent evaluates the corresponding access rules, starting with the highest priority. As soon as the condition of a rule is satisfied, the agent stops evaluating rules. For each rule which has been evaluated and the condition for which is not satisfied, the agent may decide to initiate subqueries, trying to gain additional knowledge that leads to the satisfaction of the condition:

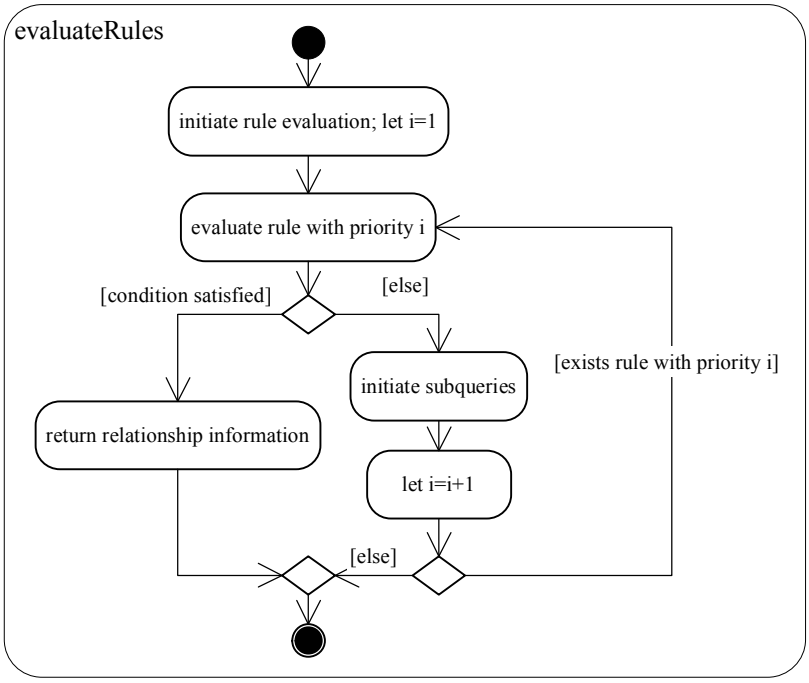


Figure 6.10: Evaluation of access rules

(a) For each chain expression in the condition, the agent formulates subqueries, if a corresponding chain cannot be found in the RI knowledge base. In order to avoid redundant communication, the agent should cache positive and negative (sub-)query results for some time.

(b) Those subqueries not on the list of open queries are initiated and put on the list of open queries.

(c) The subqueries (those which have been newly initiated and those which had already been on the list) are linked to the original query, such that on receiving a result / closing message of a subquery the agent can find the original query.

(d) The original query is put on the list of queries to answer.

3. The agent may need to initiate subqueries as described before in order to answer a query, even if the condition of the rule with the highest priority is satisfied (see below).

4. If step 2 yielded relationship information to be returned to the requesting agent, the agent returns these results to the requesting agent. If the original query has been put on the list of queries to answer, the results which have been sent to the requesting agents are linked to the original query, such that the agent can later on figure out which partial results have been already reported.

The initiation of subqueries in step 3 is especially useful if the requesting agent is an RM extension, and both agent and RM extension are acting on behalf of the same user. In general, the user agent should try to give a complete answer to the query — i.e. try to find as much relevant relationship information as possible.

If the requesting agent is another user agent, the agent the query was addressed to is free to choose if it only uses its own RI knowledge base to answer the query (and hence shows a low cooperativeness),

or if it tries to obtain additional relationship information from other agents (and hence shows a high cooperativeness). The decision between low and high cooperativeness may, for example, depend on the relationship with the person the requesting agent is acting on behalf of.

Processing of a (Partial) Result

Upon receiving a (partial) result of a subquery, the agent proceeds as follows.

1. The agent adds the result to its RI knowledge base.
2. The agent checks the queries on the list of queries to answer. If for any of those queries new (partial) results are available, which have not been sent to the respective requesting agent, the new results are submitted to the requesting agent (in compliance with the allowed usage and the agent's access rules).
3. If new results have been submitted, these are linked to the original queries.

Processing of a Closing Message

When an agent receives a closing message, it removes the corresponding query from the list of open queries. If the query, which has been closed, was the last open subquery of one or more queries on the list of queries to answer, those queries are removed from that list and closed by sending a closing message to the respective agent.

If the closing message refers to a query concerning the existence of a relationship chain, and there are no open subqueries of that query left, the query is removed from the list of processed queries, and a closing message is sent to the agent who had sent the query.

Processing of a Query for the Existence of a Relationship Chain

Upon receiving a `query-exists-chain-to-person(x, y, initiator, template, d)`-message, an agent proceeds as follows.

1. If the agent has processed this query before — i.e. a message `query-exists-chain-to-person(x, y, initiator, template, d+j)`, where $\mathtt{j} \geq 0$ — it quits processing the query.
2. If the agent is acting on behalf of the target person `y`, it closes the query:
   (a) The agent sends a `return-chain-found(query, "true")`-message to the initiator of the query.
   (b) The agent sends a closing message for the query to the requesting agent.
3. If the maximum length `d` of the chain is greater than 0, the agent generates a list of all relationship information from his user to any other user (except the user of the requesting agent), which matches the RI template `template`. The agent sends a subquery `query-exists-chain-to-person(x, y, initiator, template, d-1)` to each agent acting on behalf of one of these users.
4. If $\mathtt{d} = 0$ and the agent is not acting on behalf of `y`, the agent sends a closing message for the query to the requesting agent.
5. If the query has not been closed, the agent puts it on the list of processed queries.

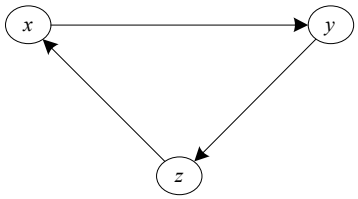


Figure 6.11: Example of a (simple) social network between three persons. Arrows indicate directed "trust" relationships.

Resolution of Access Rules

Access rules for relationship management have already been discussed in Sect. 6.5.2. Yet, strategies for checking if a condition is fulfilled have been omitted up to now. For checking if the condition of a rule is fulfilled, an agent must check all chain expressions of that rule — i.e. it must try to find a relationship chain with the respective maximum length and matching the RI template of the chain expression between the agent's user and the requesting agent's user. First, the agent may use its RI knowledge base for finding relevant relationship chains. Second, if an agent cannot decide on the basis of its RI knowledge base if a condition is fulfilled, it may pursue two different strategies for finding the missing relationship chains:

- Exploration of the network and disclosure of a relationship chain
- Checking for the pure existence of a relationship chain

Network exploration and disclosure of chain. The naive approach to checking if the condition of a rule is fulfilled is to gather as much information about the social network as possible and to search for relevant relationship chains therein. According to this approach, agents use the messages `query-RI-to-person`, `query-explore-network`, and `query-chains-to-person` in order to increase their knowledge about their users' social networks. This approach, however, only works if all agents (resp. their users) rather permit unreserved access to relationship information, because the search for relationship chains is performed locally by the requesting agent. Note that the evaluation of the fulfillment of an access rule may not even terminate in this approach:

Assume a scenario of three persons x, y, and z (for simplicity, let `x`, `y`, `z` be the URIs of x, y, and z, respectively), and a simple social network consisting of three trust relationships from x to y, y to z, and z to x (cf. Fig. 6.11). Furthermore, assume that the access rules of all three persons permit access to trust relationships only if a trust relationship chain of maximum length 5 to the requesting person exists (cf. Fig. 6.12). Let `R` be a serialization of the RDF model of the RI template R in that figure. A `query-chains-to-person(x,y,R,5)`-query directed from the agent of z to the agent of x would then result in the following message flow (for simplicity, agents are named like the persons they are acting on behalf of):

1. z sends a query `query-chains-to-person(x,y,R,4)` to x
2. x finds out that in order to allow access to the trust relationship between x and y, a trust relationship chain from x to z is needed
3. x sends a subquery `query-chains-to-person(y,z,R,4)` to y

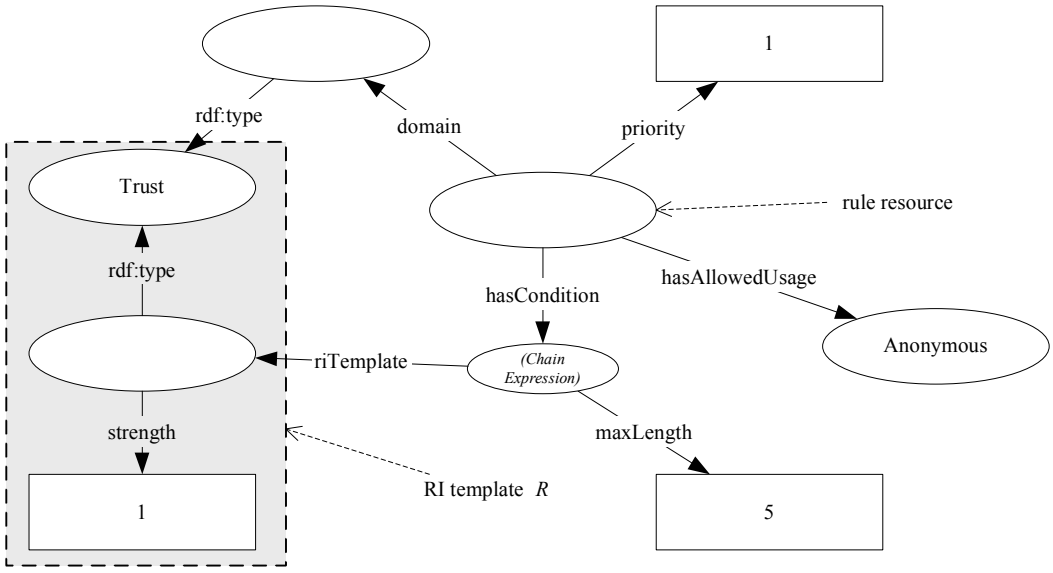


Figure 6.12: Example of an access rule allowing access to trust-relationships if a chain of maximum length 5 and strength 1 can be found between the requesting person and the owner of the information

4. y finds out that he can answer the query, but his rules only permit access to the information if a trust relationship chain to x exists
5. y initiates a subquery `query-chains-to-person(z,x,R,4)` to z
6. z finds out that he can answer the query, but his rules only permit access to the information if a trust relationship chain to y exists
7. z formulates a subquery `query-chains-to-person(x,y,R,4)` to x
8. z notices that he has sent this query to x before. z thus waits for the closure of the query instead of sending it again

The closure of the query will, however, never occur. Although in fact all conditions are fulfilled, the agents cannot find that out and thus deny access to the requested relationship information.

Check for pure existence of relationship chain. Instead of dissolving the relationship chain, this approach aims at providing a proof for the existence of a suitable relationship chain without referencing the persons along that chain. For checking the existence of a relationship chain, an agent sends a `query-exists-chain-to-person`-message to all agents, with whose user the agent's user has a matching relationship. When a `return-chain-found`-message arrives, the existence of a relationship chain is proven. Note that on receiving a single closing message, the agent may not conclude, that no chain exists. Only if all initial branches of the query have been closed and no result has been found before, the agent may conclude that no chain exists.

By incorporating digital signatures it is possible to guarantee that the `return-chain-found`-message does in fact refer to an authentic query — details on signature and security are, however, skipped in this work and left to concrete implementations of these concepts.

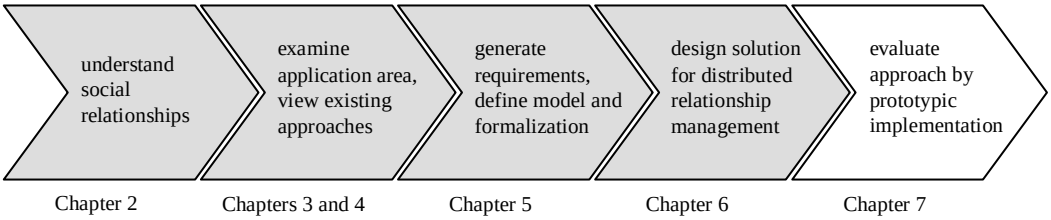


Figure 6.13: Roadmap of this thesis and intermediary results

6.7 Summary and Applications

Referring to Fig. 6.13, the main work of this thesis has now been done. Building upon the summary of sociologic concepts concerning social relationships (Chapt. 2) and the discussion of the application area (Chapts. 3 and 4), Chapter 5 has defined a formal representation of social relationships in terms of an OWL ontology vocabulary. This chapter discussed how relationship information can be managed in distributed systems by defining social protocols of software agents for relationship management.

In this approach, each user possesses a personal relationship management agent which is responsible for storing and managing relationship information on behalf of the user. By id federation, agents are linked to RM extensions of applications generating or using relationship information (cf. Sect. 6.3). Relationship management agents interact with each other in order to perform the following tasks (cf. Sect. 6.6.3):

- Exploring the social network up to a certain depth
- Checking if a relationship chain with certain characteristics from one person to another exists
- Retrieving relationship chains with certain characteristics from one user to another

When exchanging relationship information, agents consult their users' access rules in order to prevent unauthorized access to relationship information. Different from access control lists, capabilities, or role-based access control, this work takes a relationship-centric approach to access control. Access to relationship information is granted, if a corresponding access rule exists and a relationship chain matching the condition of that rule between the requesting person and the owner of the relationship information can be found (cf. Sect. 6.5.2).

The rest of this section picks up the discussion of the two scenarios presented in Sect. 1.6. The approaches developed in this work are applied to these scenarios on a conceptual level, illustrating the benefits this work yields with respect to the scenarios.

6.7.1 Supporting Users in Assessing the Reputation of Other Users

In this scenario, a user needs information about the reputation of a vendor in an online auction house or marketplace (cf. Sects. 1.3 and 1.6.1). On the basis of the relationship management multiagent system, the problems in this scenario can be solved as follows.

Basic infrastructure. The online auction house or marketplace fulfills the function of an application (the term "web platform" will be used to refer to this application in the following), as shown in Fig. 6.1. It must be equipped with an RM extension (cf. Sect. 6.4). Each user of the web platform possesses a relationship management agent.

Approach 1: Finding senior experts. This approach aims at finding a person who "knows" the vendor the user needs information about and with whom the user has a (possibly indirect) relationship.

1. On the web platform, the user requests the federation of the local user ID and the global user ID (cf. Sect. 6.3). The identity federation is established. This step has to be done only once for each application.

2. On the web platform, the user selects the vendor he needs information about. He defines a list of relationship types which should be taken into account for finding a senior expert and an upper bound for the length of relationship chains.

3. The RM extension of the web platform retrieves the global user ID of the target person. If the target user has not federated his global user ID with his local user ID, the task cannot be completed and is aborted here.

4. The RM extension sends a `query-chains-to-person(x,y,T,d)`-query to the agent acting on behalf of the user, including the relationship type and the maximum length as parameters.

5. The agent processes the query as described in Sect. 6.6.4 and sends results to the application (i.e. its RM extension). For processing the query, the agent will most probably have to initiate subqueries[15] to all other agents with whose user the user of the agent has a relationship matching the RI template `T`.

6. The RM extension visualizes the query results and presents the visualization to the user.[16]

7. If the query was successful, the user is now aware of persons in his own relationship network who know the vendor he needs information about.

Approach 2: Using weighted ratings depending on relationship chains. The second approach uses weighted ratings to calculate the cumulative rating, which depend on the "closeness" of the relationship between the issuer of the respective rating and the object of the rating. This approach assumes that the user's local user ID and his global user ID have been federated as described in Sect. 6.3. When the user requests a cumulative rating of a vendor, the web platform generates a list of all ratings the vendor has achieved on the platform. For each entry on this list, the web platform proceeds as follows.

1. If the user who issued the rating has not federated his local user ID with his global user ID, the rating is not taken into account for calculating the cumulative rating.

2. Otherwise, the RM extension of the application sends a `query-chains-to-person`-query to the agent acting on behalf of the user, where only relationships of the type `RecommenderTrust` (cf. Sect. 5.5) should be taken into account. The maximum length parameter of the query may either be a default value defined by the web platform, or chosen by the user in his general web platform preferences.

[15] Subqueries should be initiated unless the query can be answered by consulting the local relationship knowledge base.

[16] Although methods for the visualization of social networks are not addressed in this work, Fig. 7.4 (p. 137) depicts a prototypic visualization embedded in a web platform. It is likely that implementations of these approaches will also provide visualization components, which can be plugged into applications.

3. The agent processes the query as described in Sect. 6.6.4 and sends results to the application (i.e. its RM extension).

4. For each chain found, the RM extension calculates the strength of the chain. The highest strength value obtained this way is the weighting factor for the rating.

The cumulative rating is then calculated on the basis of all weighted ratings. This ensures that only "trusted" ratings are considered in the cumulative rating. The obvious drawback of this approach is performance. Calculating weights "just in time" is impractical due to the high communication overhead. Applications choosing this approach should therefore proactively search for relationship chains and cache relevant relationship information concerning their users.

6.7.2 Structuring the Personal Social Network

This scenario aims at increasing the awareness about one's own social network — i.e. assist the user to remember and find persons in his social network and find connections between two persons in his network. The results of this work can be applied to develop a social network browser. This is a special application connected to the relationship management application acting on behalf of the user, which addresses the questions raised in Sect. 1.6.2. The social network browser can especially assist the user with keeping aware of weak relationships (cf. Sect. 2.2). Useful tasks that should be fulfilled by the social network browser are the following:

- Visualize the social network of the user
- Provide exhaustive means of selecting parts of the network to visualize — e.g. on the basis of the relationship type, or the strength value
- Highlight the temporal evolution of the social network — show people moving into or moving out of the social network
- Keep relationship information up-to-date and extend the knowledge about the social network (in collaboration with special strategies for proactiveness of the user agent)

The main challenge in this scenario is the design of good user interfaces providing intuitive access to the social network and hiding the high complexity of the formalization and the agent society. This issue is, however, too complicated to be appropriately addressed in this thesis and is thus left for future work.

6.8 Comparison with Existing Approaches

This work has presented an exhaustive approach to social relationship management in internet-based communication and shared information spaces. No existing approach with a comparable scope is known to the author. Yet, there are approaches addressing small fragments of the scope of this thesis (cf. Sect. 3.6). This section aims at exposing the main advantages of this approach compared to those existing approaches.

Centrality vs. decentrality. On of the most important advantages of the approach presented in this thesis is decentrality. Different from existing approaches like Referral Web (Kautz et al., 1997b), PeCo Mediator II (Ogata et al., 2001), Social Network Fragments (Boyd, 2002), and IKNOW (Contractor et al., 1998), relationship information is stored in a decentral manner by an agent of the user's choice. This approach allows the user to retain full control over his relationship information at all times. The user is always aware what information is currently stored by his agent and may remove it, if desired.

Extensibility vs. static vocabulary. The approach of this work is highly adaptive — with respect to specific applications or application areas, additional relationship types can be defined matching the respective requirements. At the same time, specific relationship information is still understandable for other applications due to the use of a general ontology language.

In contrast, existing approaches (e.g. ContactMap (Nardi et al., 2002), IKNOW, ReferralWeb) rely on static relationship types and do not define interrelations between different types. However, this leads to a limited expressiveness.

Private vs. public sources of relationship information. Concerning private relationship information (e.g. email histories, buddylist membership, address book entries, etc.), existing approaches pursue two different strategies. Some approaches assume or require the users' consent to use private relationship information and present it to other users (e.g. PeCo Mediator II, Social Network Fragments, Visual Who (Donath, 1995)). Other approaches do not use private relationship information but only take into account public data (e.g. Referral Web, IKNOW).

This work allows the incorporation of private relationship information without requiring the users' general consent to making it public. Instead, a powerful and flexible rule-based access control is applied for the protection of private relationship information against illegitimate access. This leads to a very complete view on each user's social network with respect to internet-based communication and shared information spaces.

General representation vs. proprietary applications. Existing approaches use proprietary representations of relationships which only address those aspects needed for the application. This prevents the exchange of relationship information with identical semantics that originates from different sources. In contrast, this work has defined a general, application-independent representation of social relationships which enables the integration of relationship information from heterogeneous sources.

Chapter 7

Implementation — The InReM Framework

Building on the modeling of social relationships presented in Chapt. 5 and the conceptual presentation of the multiagent system for distributed relationship management in Chapt. 6, this chapter discusses the prototypic implementation of these concepts — the InReM framework.[1] *The InReM framework may thus be regarded as an evaluation of those concepts consisting in the proof of their practicability. The InReM framework integrates with the Cobricks community support system and can thus be integrated with any Cobricks-based community platform. Apart from integrating InReM with Cobricks, RM extensions for Email and Usenet will be sketched.*

7.1 The InReM Framework

The goal of the InReM framework[2] is to provide a prototypic implementation of the concepts developed in this thesis. It is designed to be suited both for use in a web-platform-based approach and in an agent approach:

- InReM integrates with the Cobricks Community Support System and may thus be deployed in any Cobricks-based virtual community platform (see Sect. 7.3).
- Due to its modular design, InReM may also be used in a Java-based multiagent system (see Sect. 7.2).

The InReM framework is implemented in Java and consists of six packages:

de.tum.inrem. This package contains all interfaces of the InReM framework core — i.e. those interfaces for managing relationship types, relationship information, and ontologies.

de.tum.inrem.impl. The implementations of the interfaces defined in de.tum.inrem can be found in the package de.tum.inrem.impl.

de.tum.inrem.cobricks. The de.tum.inrem.cobricks package contains all classes necessary for the integration of InReM with the Cobricks community support system — i.e. a bridge to the Cobricks user management and a wrapper for the InReM relationship management classes which can be deployed in a Cobricks-installation.

[1] I am indepted to the students Andreas Scheerer and Wolfgang Sprung, who have done important parts of the implementation on the basis of my specifications.

[2] InReM is an acronym for **In**teroperable **Re**lationship **M**anagement.

de.tum.inrem.visualization. This package contains interfaces for visualizing relationship networks.

de.tum.inrem.visualization.impl. The implementations of the visualization interfaces can be found in this package. The implementation is based on the Royere framework for graph visualization (Marshall et al., 2001) and produces SVG[3] output.

de.tum.inrem.agent. The de.tum.inrem.agent package and its subpackages contain an implementation of the concepts discussed in Chapt. 6.[4] The package provides an InReM agent, which can be run as a client/server application on any desktop PC. The agent communicates with other agents via socket connections and exchanges relationship information with them.

Figure 7.1 depicts an UML diagram of the most important interfaces in the de.tum.inrem package. The rest of this section describes the de.tum.inrem package in more detail.

7.1.1 Relationship Type Management

The prototypic InReM framework does not use the OWL vocabulary introduced in Chapt. 5 for defining relationship types, but a proprietary XML format. This is due to the fact that at the time the InReM framework was outlined and implemented, OWL support was not yet available. The Jena2 semantic web toolkit offers basic OWL support, but was not released before 2003 — at that time, most of the InReM core implementation had been already done. It is, however, possible to replace the relationship type management classes with OWL-based implementations in the future (as soon as Jena2's OWL support gets stable), without having to change much in the rest of the InReM packages. The waiving of OWL support, however, leads to a slightly reduced expressiveness of the relationship type model. Yet, all main concepts of Chapt. 5 are accounted for in the prototypic InReM implementation. Limitations mainly consist in the lack of classification mechanisms for relationship types and ontology concepts (see below).

The InReM framework defines the interface `RelationTypeManager` for managing relationship types. Implementations of this interface are responsible for retrieving type definitions, parsing them, and storing the models of relationship types. Relationship types are defined in XML documents which are retrieved from web servers via the http protocol. In Fig. 7.1, `RelationTypeServer` does not actually refer to a java interface, but to a web server offering type definitions.

Internally, relationship types are represented by instances of a class implementing the `RelationType` interface. Each such instance may in turn contain a reference to an instance of a class implementing the `ProfileRelation` interface, if the relationship type defines a profile relation and/or a strength function.

The interface `RelationTypeManager` offers methods for retrieving `RelationType` objects, checking if one relationship type is a generalization of another type, and checking the consistency of the type hierarchy (cf. Tab. 5.1).

7.1.2 Ontology Management

As mentioned above, InReM does not yet contain OWL support. Ontologies for attribute values of relationship information (e.g. the topic of a communication relationship; see also Sect. 5.1.1, p. 71) are not defined in OWL, but in RDF Schema. InReM essentially adopts the RDF Schema model for representing ontologies — i.e. there are resources and a special subset of resources called classes.

[3] http://www.w3.org/Graphics/SVG/

[4] The implementation has been done in the course of a "Systementwicklungsprojekt" on the basis of this thesis.

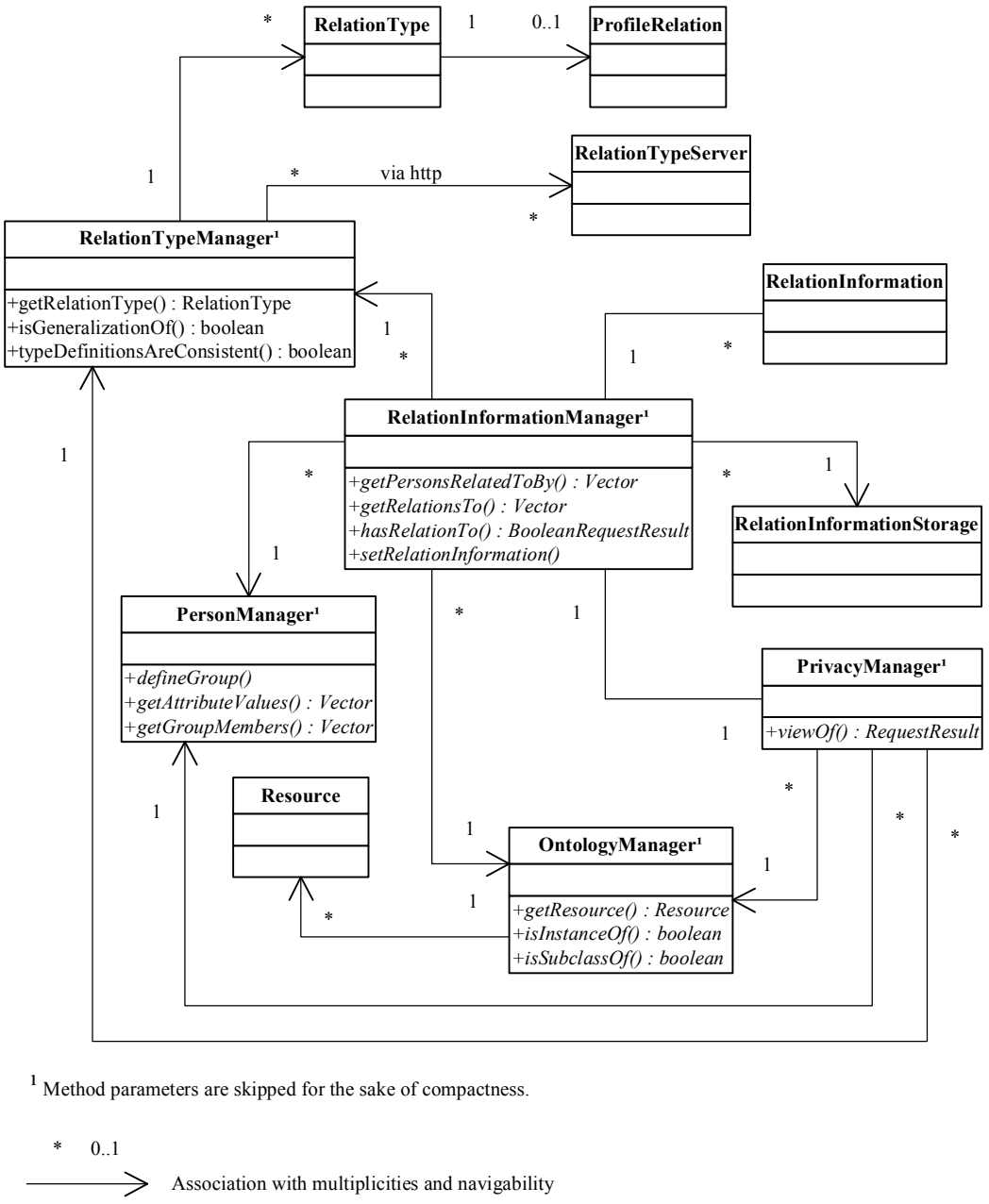


Figure 7.1: UML diagram of the main interfaces in the de.tum.inrem package. Only the most important methods are shown for each class. Methods for setting and retrieving object attributes are generally omitted.

Resources may be instances of classes and at the same time be classes themselves. The instantiation operation is indicated by the `rdf:type` property. Classes may be arranged in a subclass-hierarchy. The subclass-relation between two classes is indicated by the `rdfs:subClassOf` property.

Essentially, the `OntologyManager` interface offers methods for:

- Loading ontologies — i.e. retrieving and parsing RDF Schema documents and setting up the corresponding internal model (method `void loadOntology(String url)`)
- Retrieving a resource for a given URI
  (method `Resource getResource(String uri)`)
- Checking, if a resource is an instance of a certain class
  (method `boolean isInstanceOf(Resource individual, Resource class)`)
- Checking, if a class in a subclass of another class
  (method `boolean isSubClassOf(Resource subclass, Resource class)`)

7.1.3 Person Management

The `PersonManager` interface is responsible for managing URIs and profiles of persons. As has been discussed in Chapt. 5, groups may also be viewed as persons. Therefore, the `PersonManager` interface provides methods for explicitly defining groups by listing their members (methods `String defineGroup(String groupuri, Vector members)`, `Vector getGroupMembers(String groupuri)`).

Although the `PersonManager` interface offers a method for retrieving attribute values from the user profile concerning a specific person, it does not provide methods for setting those values. Concrete implementations of the interface must define how attribute values can be set — e.g. by using the Cobricks user management module for storing and managing user profiles.

7.1.4 Relationship Information Management

The central interface in the InReM framework is `RelationInformationManager`. Its functions are the following.

- Persistent storage of and access to `RelationInformation` objects (via an implementation of `RelationInformationStorage`) — i.e. manage a relationship knowledge base
- Add relationship information to the relationship knowledge base
  (method `void setRelationInformation(Relationinformation ri)`)
- Check if a person has a relationship of a certain type with another person
  (method `BooleanRequestResult hasRelationTo(String personURI1, String personURI2, RelationType relationType, String requestingPerson)`)
- Retrieve all relationships between two persons
  (method `Vector getRelationsTo(String personURI1, String personURI2, String requestingPerson)`)

- Retrieve all persons connected to a certain person by a relationship of a specific type (method `Vector getPersonsRelatedToBy(String personURI, RelationType relationType, String requestingPerson)`)

Implementations of `RelationInformationManager` are also responsible for calculating the extension of the relationship knowledge base (in terms of Sect. 5.4).

The `RelationInformationManager` interface is designed to be used in a centralized system — e.g. a Cobricks community support system. There is a second interface called `PersonalRelationInformationManager` offering essentially the same functions. This interface is, however, designated to be used in client software — e.g. the agents of a multiagent system for relationship management. The additional methods can be used to bind an instance of the interface to a specific person.

Concerning the `RelationInformationStorage` interface, two implementations have been done. The first one uses a mysql database[5] as storage system and is mainly targeted at a server-based solution. The second implementation takes a file-based approach and is especially suitable for relationship management agents residing on client devices.

7.1.5 Privacy Management

The `PrivacyManager` interface is responsible for rule-based access control of relationship information. For this purpose, it offers the method `RequestResult viewOf(String ownerPersonURI, String otherPersonURI, RelationInformation ri)`. This method checks if the person identified by `otherPersonURI` may access the relationship information object `ri`, which is owned by the person identified by `ownerPersonURI`.

For managing access rules, the interface provides two methods for defining and deleting rules. The current implementation of the `PrivacyManager` interface supports level 0 rules and level 1 rules (cf. Sect. 6.5.2). Figure 7.2 illustrates the interaction of an instance of `RelationInformationManager` and the associated `PrivacyManager`. When the `PrivacyManager` contacts the `RelationInformationManager` in order to check if a certain relationship between two persons exists, it uses a special method, which does not recursively invoke the `PrivacyManager` for checking access rights.

7.1.6 RM extensions

The InReM package supplies the interface `PollInterface` for accessing RM extensions of applications. An RM extension implementing this interface can be easily connected to an instance of `RelationInformationManager` by using a registration method provided by the manager. This kind of connection between an RM extension and a `RelationInformationManager` realizes the agent-triggered approach for exchanging relationship information (cf. Sect. 6.4).

7.2 InReM Agents — Distributed Relationship Management

The de.tum.inrem.agent package and its subpackages contain an implementation of a multiagent system, which is based on the concepts discussed in Chapt. 6. The main class of the package — `de.tum.inrem.agent.AgentMain` — can be started as a client/server application

[5] http://www.mysql.org

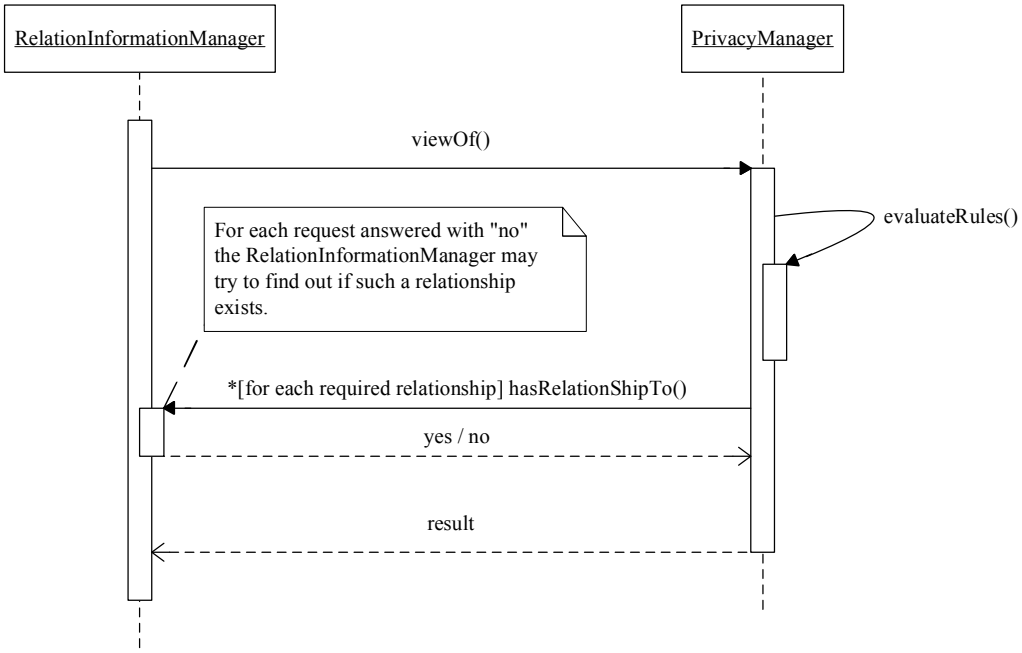


Figure 7.2: Interaction of `RelationInformationManager` and `PrivacyManager`

on a user's desktop PC. The main class initializes an InReM agent (an instance of the class `de.tum.inrem.agent.InremAgent`). Figure 7.3 depicts a screenshot of an InReM agent.

An InReM agent has the following features.

- Use of a standard LDAP server as agent registry
- Flexible binding of RM extensions (agent-triggered approach, cf. Sect. 6.4)
- Privacy management on the basis of level 1 rules
- Support of multiple query types (cf. Fig. 7.3):
  - Check, if a relationship of a certain kind exists between two persons
  - Find persons to whom a relationship of a certain kind exists
  - Explore the social network
- Social network visualization in SVG format

The de.tum.inrem.agent* packages are based on the de.tum.inrem package and the corresponding implementation. In order to keep the installation simple, the `RelationInformationStorage` interface has been implemented on the basis of the local filesystem. Hence, no local database is needed, all data is stored in the filesystem. Communication with other agents takes place via standard peer-to-peer socket connections.

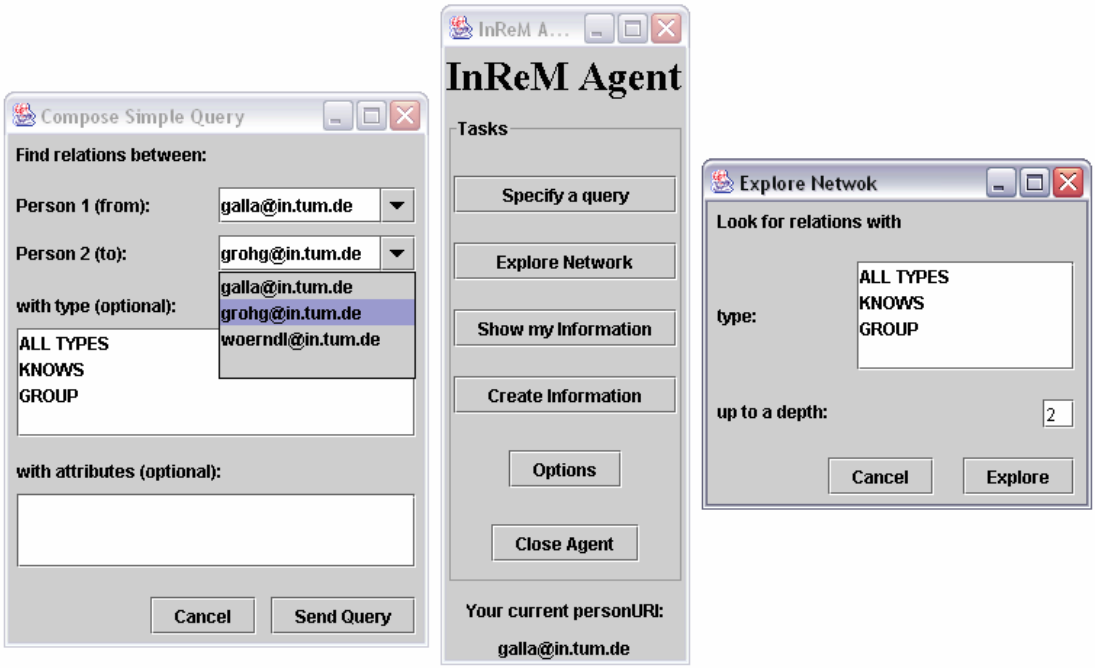


Figure 7.3: Screenshot of the prototypic InReM agent implementation. The left window contains the query interface for initiating queries for paths from one person to another person, the center window is the main window of the agent and contains the main menu, and the right window contains the dialog for exploring the social network up to a certain depth.

7.3 Prototypic Integration of InReM with Applications

The de.tum.inrem package contains interfaces for managing relationship information. Apart from these interfaces it is, however, necessary to implement RM extensions for those sources of relationship information which are to be accounted for in the InReM framework. This kind of integration of InReM with other applications has prototypically been implemented for the Cobricks community support system, IMAP mail, and Usenet news. These three approaches are briefly sketched in the remainder of this section.

Integration with Cobricks. The InReM framework has been integrated with the Cobricks community support system. The package de.tum.inrem.cobricks contains two classes implementing the integration:

- The class `CobricksRelInfoManager` implements the Cobricks manager interface and the `RelationInformationManager` interface. An object of this class can be deployed as a Cobricks manager by customizing the Cobricks configuration files. Other Cobricks components may locate and access the relationship management component just like any other component (e.g. the item management component or the user management component).
- The class `CobricksPersonManager` implements the `PersonManager` interface. For retrieving attribute values of users, it uses the Cobricks user management component.

The integration has been tested in the development system of the UnternehmerTUM community[6], which is based on Cobricks. In order to evaluate the integration, an RM extension (i.e. an implementation of the interface `PollInterface`) for the buddylist component of the UnternehmerTUM community has been implemented and connected to the relationship management component. A visualization module has been deployed using SVG as its output format. Figure 7.4 depicts a screenshot of the UnternehmerTUM development system showing a buddylist relationship network visualization.

Integration with Usenet. For integrating InReM with Usenet, a `PollInterface` implementation has been developed which analyzes selected newsgroups. The headers of usenet postings to these newsgroups are stored in a database in order to increase performance. This database also solves the problem that most usenet servers drop old messages after some time, so that those messages are not available for analysis anymore. The generation of relationship information is done as described in Sect. 3.2.3.

Integration with IMAP. For the prototypic integration of InReM with email, the IMAP protocol has been chosen. The IMAP protocol is a rather popular protocol for accessing emails. Different from the POP3 protocol, that downloads messages to a local mailbox and (usually) removes them from the server, IMAP organizes messages on the mail server in such a way that the messages can be read from any client device. Thus, an IMAP email account is a good starting point for analyzing personal email communication.

[6]The UnternehmerTUM community is located at http://www.unternehmertum.de/. It is a virtual community platform targeted at students and scientists at Technische Universität München, who are interested in entrepreneurship. The UnternehmerTUM community has been developed in the course of the TiBiD project at Technische Universität München, and is hosted by UnternehmerTUM GmbH, Garching. Its latest version was launched in September 2003.

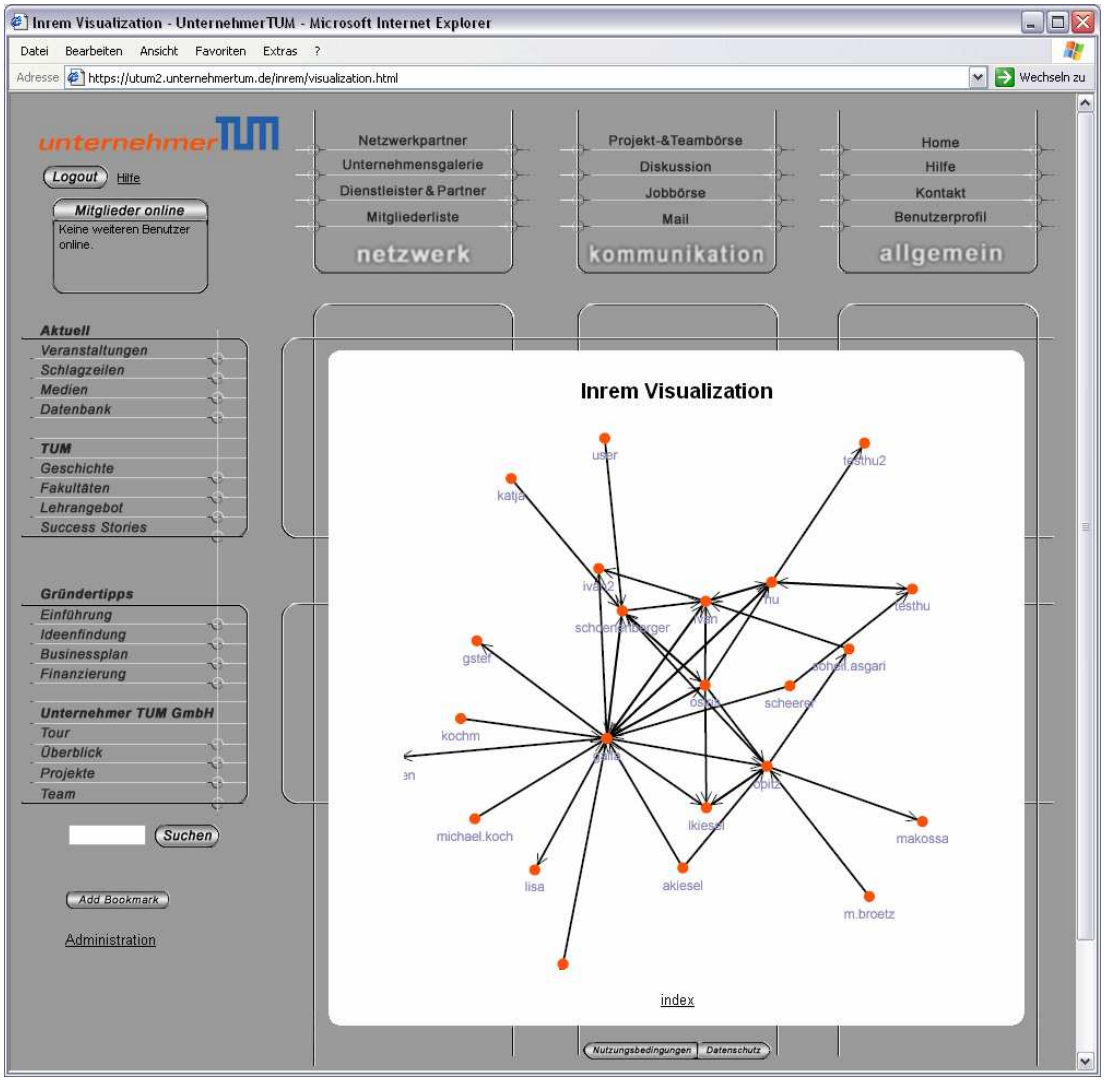


Figure 7.4: Prototypic integration of InReM with the Cobricks community support system. This screenshot was taken from the www.unternehmertum.de development system, which is based on Cobricks and uses a Cassiopeia Community Application Server as environment. The Adobe SVG plugin (http://www.adobe.com/svg/viewer/install/) — which has been used in this picture — supports zooming and panning, such that the user can interact with the presentation in order to focus on details.

In the configuration of the `IMAPPollInterface` implementation of `PollInterface`, a list of incoming folders and a list of outgoing folders must be defined. Messages in these folders are interpreted as incoming mail and outgoing mail, respectively.

7.4 Experiences with the Prototypic Implementation

First experiences with the prototypic implementation have shown the feasibility of the approach of this thesis. Of course, the validity of these results are limited and should be interpreted as "proof of concept" rather than as a thorough evaluation of the concepts developed in this thesis. A thorough validation, however, requires much larger efforts — in particular with respect to user interface design, performance, robustness and user studies. Therefore, this is out of the scope of this thesis. Yet, tests with the prototypic implementation have yielded important hints for future work, which will be discussed in the rest of this section.

One of the design goals of the InReM implementation was to provide simple programming interfaces for the relationship management system without exposing the full complexity of the relationship ontology reasoning processes. Although the current implementation does not yet use OWL for modeling relationship types but a proprietary typing mechanism based on XML documents, a similar set of interfaces can be easily implemented on the basis of OWL — as soon as stable OWL support is available. One of the benefits of providing specific interfaces for relationship management instead of exposing the full ontology to the application is that the implementation can be more efficient because it does not need OWL's full expressiveness. Reasoning algorithms can be optimized according to the specific requirements of the relationship model.

The integration with the UnternehmerTUM community[7] has been done in a centralized manner — i.e. the social relationship management module has been integrated with the Cobricks community support system and deployed on the community test server (cf. Fig. 7.4). This approach dispenses with the need for installing a relationship management client on each user's personal computer. Primarily, the integration proves the feasibility of the approach developed in this work with respect to a centralized community support system.

An important experience with respect to the integration with the UnternehmerTUM community is that users are strongly concerned about their privacy. In discussions, many people reported that they fear to loose control about who gains access to their social network. Although this problem can be dealt with by appropriately defining access rules for relationship information, advanced concepts are needed because people shy at the complexity of this process. A possible solution might be the definition of documented sets of "standard rules" with various restrictiveness. The user could choose one of those sets without needing to worry about syntactic and logic details of the rule management process. Thus, improvements concerning privacy are strongly related to good user interfaces for the relationship management system which hide the underlying the complexity of the system and at the same time do not impose functional limitations on the system. These issues are, however, left for future work, because they require large programming efforts.

Tests with the InReM agent implementation have shown the feasibility of the agent-based approach to distributed relationship management. An InReM agent is a simple java application which does not rely on additional components (such as a database). Its installation is thus very easy. Limitations of the InReM agent implementation (the development and test of which has been done in the course of a "Systementwicklungsprojekt" by a student) are robustness, the user interface — because of the limited time, only a very simple user interface could be developed which provides access to

[7] http://www.unternehmertum.de/

all important functions — and the visualization. The visualization of social networks is based on the Apache Batik library[8], which is, however, limited in some SVG language details (e.g. interactivity).

Future work concerning the InReM agent implementation should address the question of how to deal with users who are not permanently online. The current implementation just ignores offline users, leading to the social network being possibly incomplete if some users are not online while a query is solved. As has been suggested before, a solution to this problem might be to extent the agent implementation such that an agent can act on behalf of several users at the same time. Such agents could be run by third parties on a server which is permanently online and could offer a web interface for the interaction between the agent and its users.

Finally, as has been mentioned above, the InReM implementation is not yet based on OWL, because at the time the system was designed, no stable OWL library was available. Current development efforts concerning Jena2, however, include OWL support. By using Jena2, future versions of the InReM implementation can dispense with the proprietary typing mechanism for relationships and offer the full power of the relationship ontology vocabulary defined in Chapt. 5.

[8] http://xml.apache.org/batik/

Chapter 8

Conclusions and Future Prospects

This final chapter summarizes the central results of this work. Directions for future work are suggested, especially with respect to the design of user interfaces, visualization, security, and identity management. Because good visualization methods are key factors for the acceptance of implementations of the concepts developed in this thesis, promising approaches for visualizing social networks and ontologies are discussed in more detail.

8.1 Summary

Starting from virtualization tendencies in today's organizational structures and everyday life, this work has discussed two main challenges of internet-based communication systems and shared information spaces: "how can the reputation of another (not personally known) user be assessed?" and "how can people keep aware of their (online) social network?" The first challenge is especially relevant with respect to partnering in virtual organizations (B2B), online auction houses (C2C/C2B), and opinion platforms (C2C). The second challenge is important for navigating the own social network — e.g. for finding a new job or acquiring knowledge.

From these two challenges, it has been concluded that a general representation of social relationships is required. By assisting the user in finding relationship chains to unknown people, the partnering challenge can be solved. By visualization of the social network, particularly including weak relationships, navigation in the social network can be supported.

Realizing the demand for a general representation of social relationships in internet-based communication systems and shared information spaces, this thesis has discussed basic sociological notions and summarized central characteristics of social relationships (Chapt. 2).

Apart from discussing social relationships from a sociological point of view, their importance has been analyzed in the domain of internet-based communication and shared information spaces (Chapt. 3). Existing approaches to relationship management have been reviewed. The discussion of the application area has been completed by a list of relationship types occurring frequently in groupware and communityware.

Building upon the analysis of the application area and the summary of characteristics of social relationships, Chapt. 5 has introduced a formalization of social relationships based on semantic web technologies. The main design goals of the formalization have been the following:

- Fostering interoperability
- Independence from proprietary applications

- Extensibility
- Integration of privacy protection

Interoperability and extensibility are achieved by using the Web Ontology Language (OWL) for representing social relationships. OWL is part of the semantic web layer cake and builds upon RDF(S) and XML, which are targeted at providing general frameworks for representing metadata and tree-structured documents, respectively. The formalization of social relationships is thus clearly layered on top of the ontology layer of the semantic web layer cake. On the one hand, this guarantees interoperability with respect to different applications exchanging relationship information, because the semantics of OWL ontologies is well-defined. On the other hand, by using OWL, extensibility is guaranteed, because OWL includes mechanisms for importing and merging ontologies. Hence, it is not necessary to provide an overall ontology about social relationships. Starting from a basic ontology defining most common relationship types, modular ontologies can build upon these core types and specialize them, if needed.

Independence from proprietary applications is achieved by including the definition of measuring methods in type definitions. The profile relation allows for the explicit definition of measuring methods for relationships based on user profile comparison. In the absence of a profile relation, measuring methods can be provided in a natural language description. The representation of social relationships is based on general concepts originating from sociology: algebraic properties, a strength value, a profile relation, and generalizations. It is thus independent from proprietary relationship models, which usually contain only those aspects of relationships needed for the application.

Privacy protection has been addressed in connection with distributed relationship management (Chapt. 6). Different from existing approaches, this work has taken a relationship-centric approach to privacy protection. Access to relationship information is granted if certain relationship chains from the owner of the information to the requesting person can be found. The kind of relationship chains required is defined by the user in a set of rules. This approach allows for a higher flexibility and dispenses with additional artifacts like access tickets or roles.

Building upon the formalization of social relationships, this work has designed a multiagent system for distributed relationship management. Agents act on behalf of one or several persons and exchange relationship information in order to answer queries initiated by their users or by applications. Three query types can be distinguished:

- Exploring the social network up to a certain depth
- Checking if a relationship chain with certain characteristics from one person to another exists
- Retrieving relationship chains with certain characteristics from one user to another

On the basis of these query types, solutions for the two challenges mentioned above have been described.

Finally, Chapt. 7 presented a prototypic implementation of the concepts developed in this work. An integration with the Cobricks community support system has been sketched as well as RM extensions for buddy lists, Usenet news, and IMAP mail.

8.2 Future Prospects

This thesis has presented a general approach to social relationship management in internet-based communication systems and shared information spaces. In order to present the core ideas of this

approach as clearly as possible, some aspects have been disregarded. The most important among these aspects are:

- User interfaces
- Security
- Anonymity and Identity/Pseudonymity Management

The remainder of this section briefly discusses each of these aspects, presenting suggestions for future work.

8.2.1 User Interfaces for Relationship Management

Relationship management is a highly complex task. Relationship management applications must deal with the complexity of relationship information and exchange mechanisms on the application level. At the same time, they must have simple and intuitive user interfaces, because usability is a key factor for applications to be widely accepted. User interface design for relationship management has to cope with three dimensions of complexity:

- Social networks are often large and densely knit. The visualization must be able to cope with dense graphs in an effective and efficient way and it must reduce the graph's complexity. Users should be able to browse the social network and focus relationships of special types.
- The visualization should hide the complexity of the underlying ontologies. The main challenge here is to develop appropriate means for ontology visualization (both for the relationship ontology and for ontologies referenced in values of additional attributes).
- The user interface must find intuitive ways for displaying and editing access rules. Research in the COSMOS project[1] at Technische Universität München has shown that people do not adopt complex rule systems very easily. Reducing the complexity of such systems is a major challenge for user interface development.

Rule-based systems for access control are increasingly gaining importance especially for exchanging user-related information on the internet.[2] It is thus likely that the problem of developing intuitive and at the same time powerful user interfaces for rule management will be addressed in future work.

The visualization of large graphs has been a common topic for some time. This thesis will not try to give an overview about the vast field of graph visualization but highlight some promising directions for visualizing and navigating large graphs and ontologies "in a nutshell".

Drawing and Navigating Large Graphs

Social networks can be visualized as graphs in an obvious manner by drawing persons as nodes and relationships as (possibly annotated and/or colored) edges. How to layout a graph is, however, not a trivial question (see Herman et al., 2000, for an excellent overview about graph visualization and navigation techniques). One important criterion to judge a good graph visualization is the compliance with *aesthetic* guidelines (Herman et al., 2000):

[1] These results have not yet been published and thus have an informal character.

[2] Wörndl (2003), for example, discusses privacy in distributed user profile management. His approach is based on the negotiation of access tickets building upon a set of rules defined by the user.

- Nodes and edges should be evenly distributed.
- Edges should have about the same length.
- Isomorphic substructures should be displayed in the same manner.
- The number of edge-crossings should be minimal (this raises the question of how to find out if graphs are planar).

Some of these criteria, however, are to a certain degree dependent on the type of graph which is visualized or on the layout method. If clustering methods are used for the graph layout, for example, the requirement that nodes and edges should be evenly distributed may be relaxed to some extent.

Another important aspect is the *predictability* of layout algorithms. Predictability means that two runs on the same or very similar graph should lead to identical or very similar layouts. Especially for social network visualization, predictability is an inevitable feature of the layout algorithm. If non-predictable layout algorithms were used, even small changes to the network might lead to a radically different layout and to the user's complete disorientation, because the "mental map" previously acquired would be completely invalid.

If a relationship management agent traces relationships up to a depth greater than 2 or 3, the resulting social network can be quite large. Static visualization of large social networks may quickly prove infeasible, because the information might simply not fit on the screen or at least become unreadable. Methods for navigating such graphs are thus necessary. Traditional approaches for graph navigation are zoom and pan (Herman et al., 2000). Panning means that the area of the graph which is displayed is changed. Zooming means that certain regions of the graph are displayed on a more detailed level. Two variants of zoom can be distinguished: *geometric zoom* simply "blows up" the graph (without altering its content/structure) such that more details become visible. *Semantic zoom* means that the content of the graph is adopted according to the zoom level (e.g. in a clustered graph, on a high level, clusters might be collapsed to one node). Figure 8.1 depicts the two zooming variants.

Semantic zoom is strongly connected to *graph clustering*. Graph clustering refers to clustering the set of nodes into (ideally) disjoint sets of nodes. These clusters can either be used for reducing the complexity of the graph, as in Fig. 8.1, or for advanced layout algorithms: recursive clustering of a graph yields a hierarchic structure of the nodes which may be used as input for tree-based layout algorithms (Herman et al., 2000).

A classical method for combining zooming and panning is *fisheye distortion*. Fisheye views are applied to a given graph layout and imitate the optical distortion of the view as if the user was looking through a fisheye lens. The effect of the distortion is that focal areas are enlarged, whereas the periphery is shown with less detail. Figure 8.2 shows an example of a graph layout to which a fisheye distortion has been applied. A second approach to zooming and panning is *hyperbolic graph layout*, which will be explained below.

Ontology visualization

Apart from the visualization of social networks, the visualization of ontologies is a challenge. Whereas the relationship type ontology is essentially a hierarchy, domain-dependent topics referenced in additional attributes of relationships can be quite complex.

The subclass-relationship, which is available in most ontology languages including OWL, defines a hierarchy on the classes of an ontology. Although it is usually not required that this hierarchy is a tree (OWL, for instance, allows a class to have more than one superclass), it is often "somewhat

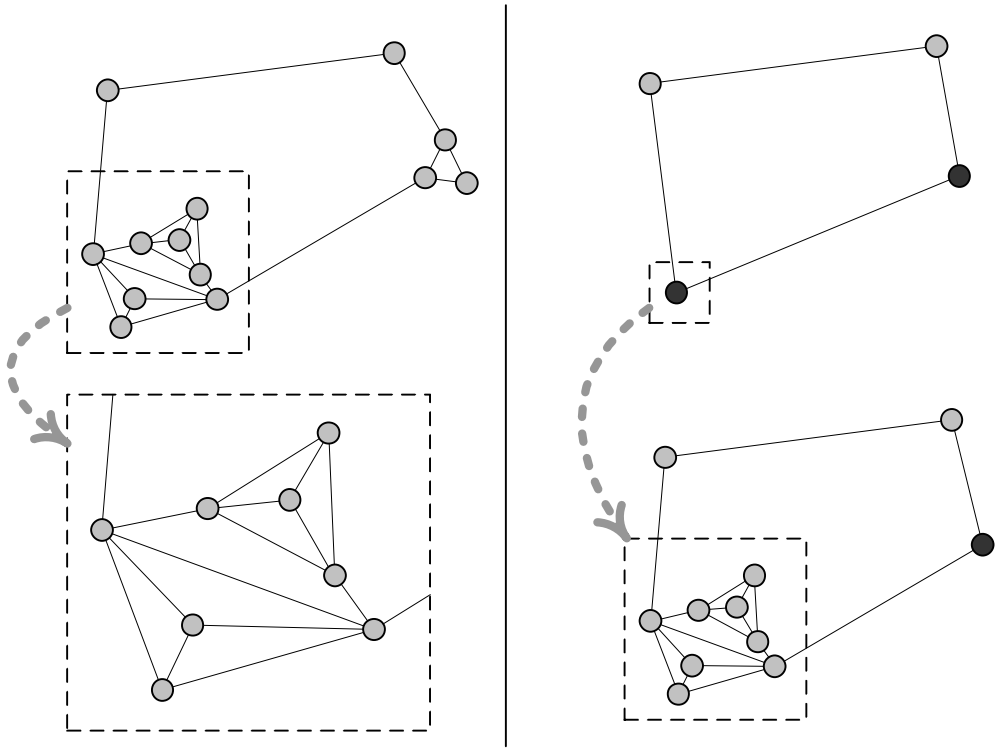


Figure 8.1: Geometric zoom (left) and semantic zoom (right)

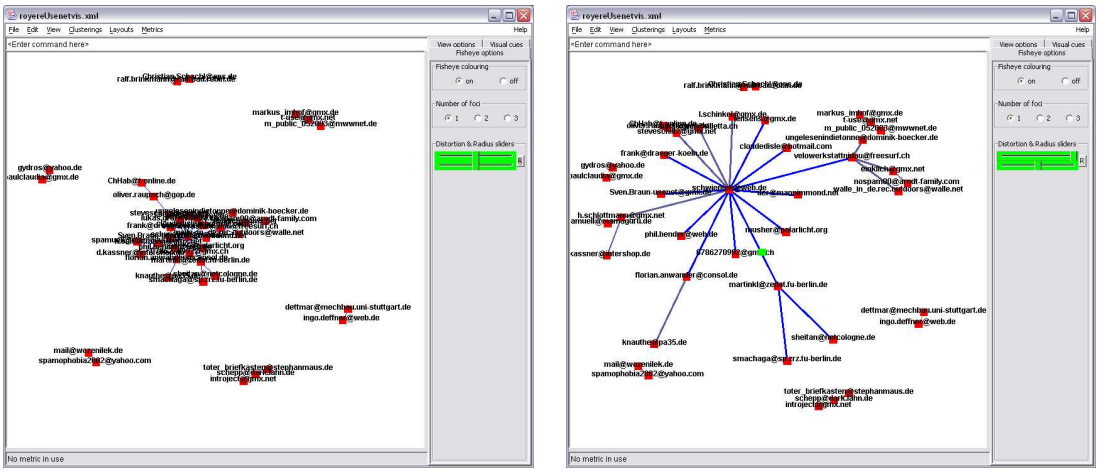


Figure 8.2: Fisheye distortion. The left screenshot shows the original graph layout. Obviously, the visualization is of little use due to the overlapping of node labels and nodes themselves. The right screenshot depicts the effect of a fisheye view on the same graph. The center cluster is shown with more detail. The periphery, however, is still visible. Both visualizations have been produced with the Royere toolkit (Marshall et al., 2001) on the basis of the same usenet data that has been used for Fig. 3.3.

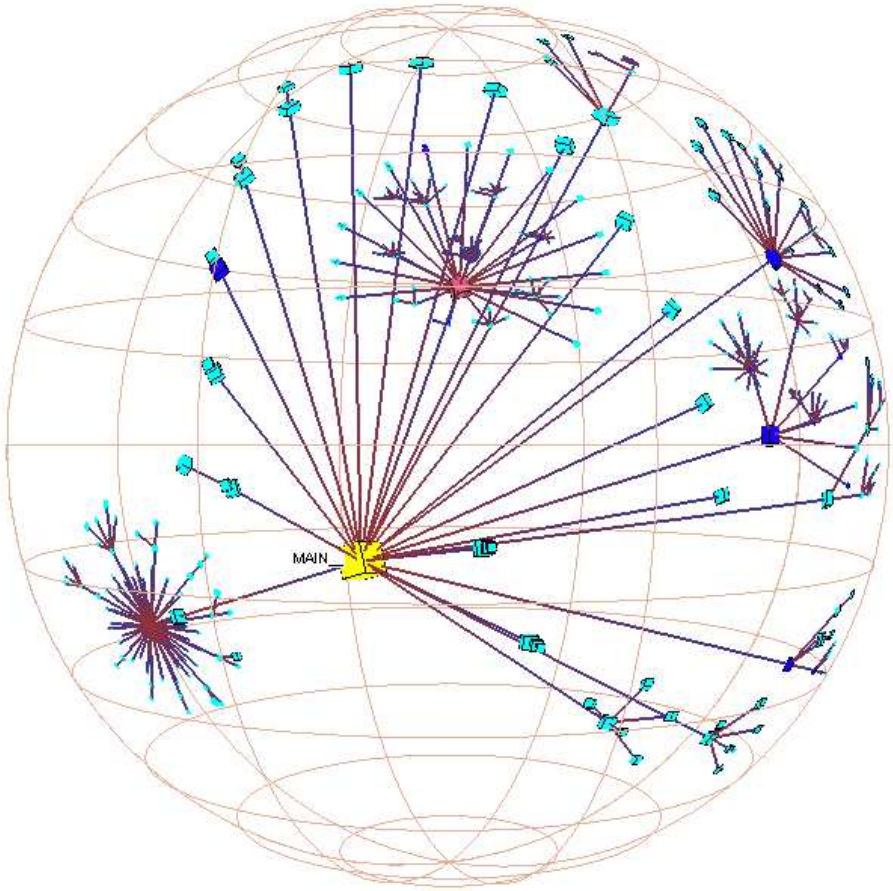


Figure 8.3: Hyperbolic tree visualization in 3D. This figure was created with H3Viewer (from http://graphics.stanford.edu/∼munzner/h3/).

like a tree" — i.e. a tree with comparatively few additional non-tree edges. *Hyperbolic layouts* have turned out to be especially suitable for tree visualization (Lamping et al., 1995). Munzner (1998), for instance, suggests a 3D layout for trees and tree-like graphs, which allows the user to rotate the graph and zoom in on areas of interests (see also Fig. 8.3). Hyperbolic layout algorithms do not use Euclidean space but hyperbolic space. In hyperbolic space, to a given (infinite) line l and a given point p not on that line, there exist *more than* one (infinite) lines $r_1, ..., r_n$ such that l is parallel to each of $r_1, ..., r_n$ (this does *not* imply that $r_1, ..., r_n$ are pairwise parallel), and $p \in r_i, 1 \leq i \leq n$ (in Euclidean geometry, there is exactly one such line). The result of the layout algorithm is then mapped to a Euclidean model of the hyperbolic space. In essence, in hyperbolic space there is simply more room so that that the distance (in the hyperbolic space) between parents, children, and siblings is approximately the same everywhere in the tree. Herman et al. (2000) present some more details on this procedure. For a detailed mathematical theory, see Coxeter (1961).

Although hyperbolic graph layout is good for visualizing the tree aspects of an ontology, a lot of information contained in an ontology is lost. For instance, it is usually not visible if two classes have common members. Fluit et al. (2003) propose a different (2D) approach, which aims at displaying such relationships between classes. Their basic ontology model is a simple superclass-subclass hi-

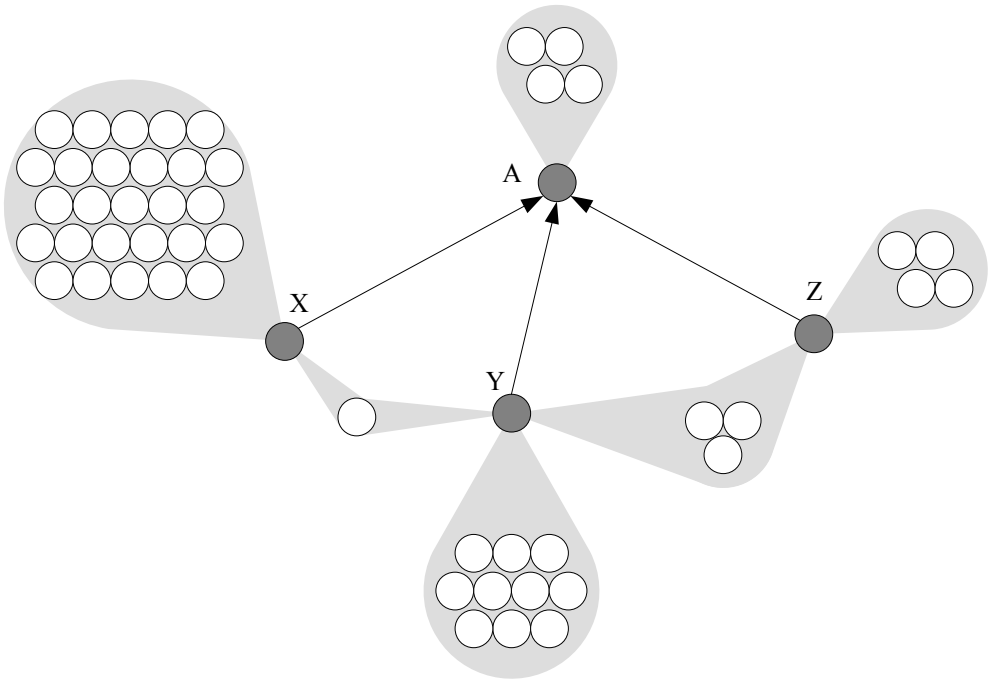


Figure 8.4: An example cluster map, according to Fluit et al. (2003). The three classes X, Y, and Z are subclasses of the class A (indicated by the arrows). Classes X and Y are connected by an edge indicating that X and Y have one member in common. Similarly, Y and Z have three members in common. The "bubbles" attached to X, Y, and Z indicate the number of members of X, Y, and Z, respectively, not including those members shared with other classes. The "bubble" attached to class A indicates that there are members of A not belonging to one of A's subclasses. Since members are *clustered* according to class membership, this approach is called a *cluster map*.The layout algorithm for positioning nodes and clusters is spring-based.

---

erarchy, where each class has a number of members. The main idea is to display additional edges indicating the number of members two classes have in common. Figure 8.4 sketches a sample visualization.

8.2.2 Security

The focus of this work has been the modeling and the exchange of relationship information. Security, however, has not been addressed. Relationship information is personal data of great value. Implementations of the concepts developed in this work should therefore incorporate security technologies:

Authentication. The multiagent system presented in Chapt. 6 relies on the mapping between agents and persons, which is maintained by the user agent registry (cf. Fig. 6.7). It is, however, important that this mapping is *trustworthy* — i.e. that the mapping is certified by a trusted authority. Otherwise, an agent could register for an arbitrary person and gain relationship information it is not authorized for. For these tasks, standard public key infrastructures on the basis of X.509 certificates[3] can be applied.

[3] http://www.ietf.org/html.charters/pkix-charter.html

Digital Signature. Authentication is closely related to digital signature, because digital signatures can be used for certifying the authenticity of documents/statements. In general, digital signatures should be attached to all messages exchanged in the multiagent system. An agent can be sure that a message m of another agent x who pretends to be acting on behalf of person p is authentic if m is digitally signed by p and the agent ID of x is included in m.

Cryptography. Along with digital signature messages should be encrypted (e.g. by using SSL[4]), so that agent communication cannot be eavesdropped by third parties.

8.2.3 Anonymity and Identity/Pseudonymity Management

One basic assumption in this work has been that the mapping between persons and IDs is public. Consequently, any message sent in the multiagent system can not only be attributed to the agent who sent it but also to the person the agent is acting on behalf of. People are, however, increasingly concerned about which aspects of their identity are visible online. Thus, it would be interesting to extend the solution developed in this work by incorporating anonymization technologies.

Another aspect is identity management: many people tend to have several identities each representing a special role. A common example is the differentiation between a business identity and a private identity. Relationship information, however, often cannot be clearly attributed to only one of these identities, e.g. there might be persons both in the business network and in the private network. The approach of this work assumes that all these different identities of a person are publicly linked. It would thus be interesting to extend the ideas of this work by incorporating mechanisms for identity management, which still allow for the combination of relationship information linked to the different identities of a person without publicly linking those identities.

8.2.4 Concluding Remarks

Social relationship management is an integral part of our lives. As more and more activities move online, data about large parts of our social networks is stored online. This thesis has shown that an expressive combination of such social network data is feasible in the domain of internet-based communication and shared information spaces. Hopefully, the results of this work will provide new impulses for future systems supporting partnering and knowledge management.

Although expressive relationship management is generally feasible, some work remains to be done with respect to ontology management. Current implementations of semantic web technologies must be considered research prototypes rather than suitable solutions for professional end-user applications. Yet, there are promising projects, such as the Jena2 semantic web toolkit (cf. Sect.4.4.5).

Much work must still be done with respect to user interfaces and visualization. This does, however, not only apply to this work, but also to semantic web technologies and social network applications in general. Some promising approaches have been presented in this work. Yet, apart from few exceptions, these are mainly research prototypes, too.

The technical feasibility of the ideas presented in this thesis does not necessarily imply their general acceptance by the user. Therefore, future research should investigate user interaction with implementations of these concepts. Such research, however, should not be done on the basis of a research prototype. Research prototypes often lack professional user interfaces and are thus often not accepted by end-users. A poorly-designed user interface, which fails to hide the complexity of

[4]Secure Sockets Layer (SSL) is a general standard for authentication and encryption between clients and servers on the internet (see http://developer.netscape.com/docs/manuals/security/sslin/index.htm).

the underlying technologies, will most likely cause a user to reject the technology without having a chance to get to know the technology's benefits. Thus, user interfaces are a pivotal aspect concerning the technology's success or failure. Due to the immense effort necessary for developing really good user interfaces, studies of the acceptance of relationship management technology could not be done within the scope of this thesis. The author assumes that future work will fill this gap and will be able to suggest improvements of the ideas presented in this thesis.

The field of social relationship management in internet-based applications is not well explored. This work has tried to combine existing approaches and at the same time draw a "big" picture of social relationship management in internet-based communication and shared information spaces. This work can serve as a building block for future approaches addressing similar issues.

Appendix A

The Relationship Ontology

```
<!DOCTYPE owl [
    <!ENTITY xsd   "http://www.w3.org/2001/XMLSchema#">
    <!ENTITY owl   "http://www.w3.org/2002/07/owl#">
    <!ENTITY rdf   "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <!ENTITY rdfs  "http://www.w3.org/2000/01/rdf-schema#">
    <!ENTITY inrem "http://www11.in.tum.de/projects/inrem/ns/inrem-ns#">
    ]>

<rdf:RDF
    xmlns="http://www11.in.tum.de/projects/inrem/ns/inrem-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

<owl:Ontology rdf:about="">
</owl:Ontology>

<owl:Class rdf:ID="Relationship">
  <rdfs:label xml:lang="en">Generic Relationship</rdfs:label>
  <rdfs:comment xml:lang="en">
    This is the generic relationship type all other types are
    specializations of. Relationships of this type are not assumed
    to have any special meaning.
  </rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#from" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#to" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
```

```
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#strength" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#normalizedStrength" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasStrengthSource" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#requiresTransitivityMeasure" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasProfileRelation" />
      <owl:maxCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTimestamp" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Person" />

<owl:Class rdf:ID="Group">
  <rdfs:subClassOf rdf:resource="#Person" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAnchor" />
      <owl:cardinality
```

```
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#definedBy" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasAnchor">
  <rdfs:domain rdf:resource="#Group" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="definedBy">
  <rdfs:domain rdf:resource="#Group" />
  <rdfs:range rdf:resource="#Relationship" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="from">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="to">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="#Person" />
</owl:ObjectProperty>

<owl:Class rdf:ID="AlgebraicProperty" />

<owl:Thing rdf:ID="Symmetric" />
<owl:Thing rdf:about="#Symmetric">
  <rdf:type rdf:resource="#AlgebraicProperty" />
</owl:Thing>

<owl:Thing rdf:ID="Antisymmetric" />
<owl:Thing rdf:about="#Antisymmetric">
  <rdf:type rdf:resource="#AlgebraicProperty" />
</owl:Thing>

<owl:Thing rdf:ID="Asymmetric" />
<owl:Thing rdf:about="#Asymmetric">
  <rdf:type rdf:resource="#AlgebraicProperty" />
</owl:Thing>

<owl:Thing rdf:ID="Transitive" />
<owl:Thing rdf:about="#Transitive">
  <rdf:type rdf:resource="#AlgebraicProperty" />
</owl:Thing>
```

```
<owl:ObjectProperty rdf:ID="hasAlgebraicProperty">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="#AlgebraicProperty" />
</owl:ObjectProperty>

<owl:Class rdf:ID="SymmetricRelationship">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAlgebraicProperty" />
      <owl:hasValue rdf:resource="#Symmetric" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#AntisymmetricRelationship" />
</owl:Class>

<owl:Class rdf:ID="AntisymmetricRelationship">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAlgebraicProperty" />
      <owl:hasValue rdf:resource="#Antisymmetric" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#SymmetricRelationship" />
</owl:Class>

<owl:Class rdf:ID="AsymmetricRelationship">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAlgebraicProperty" />
      <owl:hasValue rdf:resource="#Asymmetric" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#AntisymmetricRelationship" />
  <owl:disjointWith rdf:resource="#SymmetricRelationship" />
</owl:Class>

<owl:Class rdf:ID="TransitiveRelationship">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAlgebraicProperty" />
      <owl:hasValue rdf:resource="#Transitive" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty rdf:ID="strength">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;decimal" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="normalizedStrength">
  <rdfs:domain rdf:resource="#Relationship" />
```

```
  <rdfs:range rdf:resource="&xsd;boolean" />
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="hasStrengthSource">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="#StrengthSource" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="requiresTransitivityMeasure">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="#TransitivityMeasure" />
</owl:ObjectProperty>

<owl:Class rdf:ID="StrengthSource" />

<owl:Thing rdf:ID="Explicit" />
<owl:Thing rdf:about="#Explicit">
  <rdf:type rdf:resource="#StrengthSource" />
</owl:Thing>

<owl:Class rdf:ID="StrengthFunction">
  <rdfs:subClassOf rdf:resource="#StrengthSource" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasStrengthDefinition" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="TransitivityMeasure">
  <rdfs:subClassOf rdf:resource="#StrengthSource" />
</owl:Class>

<owl:Thing rdf:ID="Min" />
<owl:Thing rdf:about="#Min">
  <rdf:type rdf:resource="#TransitivityMeasure" />
</owl:Thing>

<owl:Thing rdf:ID="Length" />
<owl:Thing rdf:about="#Length">
  <rdf:type rdf:resource="#TransitivityMeasure" />
</owl:Thing>

<owl:Thing rdf:ID="Product" />
<owl:Thing rdf:about="#Product">
  <rdf:type rdf:resource="#TransitivityMeasure" />
</owl:Thing>

<owl:DatatypeProperty rdf:ID="hasStrengthDefinition">
  <rdfs:domain rdf:resource="#StrengthFunction" />
  <rdfs:range rdf:resource="&xsd;string" />
```

```
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasProfileRelation">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasTimestamp">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;datetime" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="hasExpirationDate">
  <rdfs:domain rdf:resource="#Relationship" />
  <rdfs:range rdf:resource="&xsd;datetime" />
</owl:DatatypeProperty>

<!-- End of basic vocabulary -->
<!-- Start of Examples -->

<owl:Class rdf:ID="Acquaintance">
  <rdfs:subClassOf rdf:resource="#Relationship" />
  <rdfs:label xml:lang="en">Acquaintance</rdfs:label>
  <rdfs:comment xml:lang="en">
    This general relationship type represents the fact that
    a person knows the other. It does neither imply any
    evaluations of the other person, nor a face-to-face contact
    between the two persons.
  </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="Communication">
  <rdfs:subClassOf rdf:resource="#Acquaintance" />
  <rdfs:subClassOf rdf:resource="#SymmetricRelationship" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTopic" />
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label xml:lang="en">Communication</rdfs:label>
  <rdfs:comment xml:lang="en">
    Relationships of this type represent all kinds of communication.
    Communication relationships are assumed to be symmetric.
    The property hasTopic indicates the topics of the relationship
    as a set of string literals. Apply the property separately for
    each keyword. Supply at least one keyword for each relationship.
  </rdfs:comment>
</owl:Class>

<owl:DatatypeProperty rdf:ID="hasTopic">
  <rdfs:domain rdf:resource="#Communication" />
```

```
  <rdfs:range  rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:Class rdf:ID="EmailCommunication">
  <rdfs:subClassOf rdf:resource="#Communication" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#strength" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#normalizedStrength" />
      <owl:hasValue rdf:datatype="&xsd;boolean">true</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasStrengthSource" />
      <owl:hasValue rdf:resource="#Explicit" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label xml:lang="en">Symmetric email-communication</rdfs:label>
  <rdfs:comment xml:lang="en">
    This type represents special communication relationships by
    specifying email as the medium of the communication.
    Relationships of this type possess a normalized strength
    value as defined by the following formula:
    Let n12(k) be the number of emails sent (via to, cc, or bcc)
    in period k from person 1 to person 2, and n21(k) the number of
    emails sent in period k from 2 to 1.
    Period 1 is the current day, period 2 is the day
    before, and so on. Let S1(k) and S2(k) be the number of
    messages sent to any person by 1 and 2, respectively, in
    period k, and R1(k), R2(k) the number of messages received from
    any person in period k by 1 and 2, respectively.
    Let w(k)=(366-k)/(133590). Then, the strength s is defined as
    1/2 * sum(k=1,...,365) w(k) * ( n12(k)/S1(k) + n21/R1(k)
    + n21(k)/S2(k) + n12/R2(k) ). Periods in which any
    of the denominators is 0 or for which no data is available
    must be skipped in the sum.
  </rdfs:comment>
</owl:Class>

</rdf:RDF>
```

Appendix B

The Rule Vocabulary

```
<!DOCTYPE owl [
    <!ENTITY xsd   "http://www.w3.org/2001/XMLSchema#">
    <!ENTITY owl   "http://www.w3.org/2002/07/owl#">
    <!ENTITY rdf   "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <!ENTITY rdfs  "http://www.w3.org/2000/01/rdf-schema#">
    <!ENTITY inrem "http://www11.in.tum.de/proj/inrem/ns/inrem-ns#">
    ]>

<rdf:RDF
    xmlns="http://www11.in.tum.de/proj/inrem/ns/rule-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:inrem="http://www11.in.tum.de/proj/inrem/ns/inrem-ns#">

<owl:Ontology rdf:about="">
  <owl:imports
    rdf:resource="http://www11.in.tum.de/proj/inrem/ns/inrem-ns"/>
</owl:Ontology>

<owl:Class rdf:ID="Rule">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#priority" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#domain" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty rdf:resource="#hasAllowedUsage" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="ChainExpression">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#riTemplate" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#maxLength" />
      <owl:cardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="AllowedUsage">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#Public"/>
    <owl:Thing rdf:about="#Anonymous"/>
    <owl:Thing rdf:about="#Private"/>
  </owl:oneOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="domain">
  <rdfs:domain rdf:resource="#Rule" />
  <rdfs:range  rdf:resource="&inrem;Relationship" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasCondition">
  <rdfs:domain rdf:resource="#Rule" />
  <rdfs:range  rdf:resource="#ChainExpression" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="riTemplate">
  <rdfs:domain rdf:resource="#ChainExpression" />
  <rdfs:range  rdf:resource="&inrem;Relationship" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasAllowedUsage">
  <rdfs:domain rdf:resource="#Rule" />
  <rdfs:range  rdf:resource="#AllowedUsage" />
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="priority">
```

```
  <rdfs:domain rdf:resource="#Rule" />
  <rdfs:range  rdf:resource="&xsd;nonegativeInteger" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="maxLength">
  <rdfs:domain rdf:resource="#ChainExpression" />
  <rdfs:range  rdf:resource="&xsd;nonegativeInteger" />
</owl:DatatypeProperty>

</rdf:RDF>
```

Bibliography

Alfarez Abdul-Rahman and Stephen Hailes. Relying on trust to find reliable information. In *Proceedings 1999 International Syposium on Databes, Web and Cooperative Systems (DWACOS'99)*, Baden-Baden, Germany, August 1999.

Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence*, number 2174 in Lecture Notes in Computer Science, pages 396–408. Springer, Vienna, September 2001.

Dave Beckett, editor. *RDF/XML Syntax Specification (Revised)*. 2003. W3C Working Draft 05 September 2003, http://www.w3.org/TR/2003/WD-rdf-syntax-grammar-20030905, work in progress.

Mary Behr. All eyes are on you. *Popular Science*, 2002.

T. Berners-Lee, R. Fielding, U. C. Irvine, and L. Masinter. Uniform resource identifiers (URI): generic syntax (RFC 2396). http://www.faqs.org/rfcs/rfc2396.html, August 1998.

Tim Berners-Lee. Semantic Web — XML2000. Talk, slides available at http://www.w3.org/2000/Talks/1206-xml2k-tbl/, 2000.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

Paul V. Biron and Ashok Malhotra, editors. *XML Schema Part 2: Datatypes*. 2001. W3C Recommendation 02 May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 164–173, May 1996.

Uwe M. Borghoff and Johann H. Schlichter. *Computer-Supported Cooperative Work: Introduction to Distributed Applications.* Springer, Berlin, Heidelberg, New York, Barcelona, Hong Kong, London, Milan, Paris, Singapore, Tokyo, 2000.

Danah Boyd. Faceted id/entity: Managing representation in a digital world. Master's thesis, Massachusetts Institute of Technology, September 2002.

Jeffrey M. Bradshaw. An introduction to software agents. In Jeffrey M. Bradshaw, editor, *Software Agents*, pages 3–46, Menlo Park, California; Cambridge, Massachusetts; London, England, 1997. AAAI Press / The MIT Press.

Martin Bürger. *Unterstützung von Awareness bei der Gruppenarbeit mit gemeinsamen Arbeitsbereichen*. Utz, Wiss., München, 1999.

Dan Brickley and R. V. Guha, editors. *RDF Vocabulary Description Language 1.0: RDF Schema*. 2003. W3C Working Draft 10 October 2003, http://www.w3.org/TR/2003/WD-rdf-schema-20031010/, work in progress.

Matt Cain. Instant messaging market development. META Group — Web & Collaboration Strategies — WCS 1230, February 2003.

Linda Carotenuto, William Etienne, Michael Fontaine, Jessica Friedman, Helene Newberg, Michael Muller, Matthew Simpson, Jason Slusher, and Kenneth Stevenson. Communityspace: Toward flexible support for voluntary knowledge communities. In *Proceedings of the Workshop "Changing Places" on Workspace Models for Collaboration*, London, 1999.

Roger Clarke. Internet privacy concerns confirm the case for intervention. *Communications of the ACM*, 42(2), February 1999.

Noshir S. Contractor, Dan Zink, and Mike Chan. IKNOW: A tool to assist and study the creation, maintenance, and dissolution of knowledge networks. In Toru Ishida, editor, *Community Computing and Support Systems*, Lecture Notes in Computer Science 1519, pages 201–217. Springer, Berlin, 1998.

George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems — Concepts and Design (Third edition)*. Addison-Wesley, Harlow, (England), London, New York, Reading (Massachusetts), San Francisco, Toronto, Don Mills (Ontario), Sydney, Tokio, Singapore, Hong Kong, Seoul, Taipei, Cape Town, Madrid, Mexico City, Amsterdam, Munich, Paris, Milan, 2001.

H. S. M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, Inc., New York, London, 1961.

Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification. W3C Recommendation 16 April 2002, http://www.w3.org/TR/P3P/, April 2002.

Christopher Creutzig, Andreas Buhl, and Phil Zimmermann. *PGP Pretty Good Privacy — Der Briefumschlag für Ihre elektronische Post*. Art d'Ameublement, Bielefeld, 1999.

D. Crocker and P. Overell. Augmented BNF for syntax specifications: ABNF (RFC 2616). http://www.faqs.org/rfcs/rfc2234.html, November 1997.

David H. Crocker. Standard for the format of arpa internet text messages (RFC 822). http://www.faqs.org/rfcs/rfc822.html, August 1982.

John Davies, Dieter Fensel, and Frank van Harmelen. Introduction. In John Davies, Dieter Fensel, and Frank van Harmelen, editors, *Towards the Semantic Web — Ontology-Driven Knowledge Management*, pages 1–9. John Wiley & Sons Ltd, Chichester, England, 2003.

Mike Dean and Guus Schreiber, editors. *OWL Web Ontology Language Reference*. 2003. W3C Working Draft 31 March 2003, http://www.w3.org/TR/2003/WD-owl-ref-20030331/, work in progress.

Chrysantos Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, Minneapolis, MN, October 17–20 2000.

Andreas Dieberger and Mark Guzdial. Cowebs — experiences with collaborative web spaces. In Lueg and Fisher (2003), pages 155–166.

Rainer Dollase. *Soziometrische Techniken — Techniken der Erfassung und Analyse zwischenmenschlicher Beziehungen in Gruppen*. Beltz, Weinheim (u. a.), 1976.

Rainer Dollase. Soziometrie. In *Psychologie des 20. Jahrhunderts, Band VII: Levin und die Folgen*. Kindler, Zürich, 1979.

Judith Donath. Visual who: Animating the affinities and activities of an electronic community. In *Proceedings of ACM Multimedia '95*, San Francisco, CA, November 5–9 1995.

Judith Donath. A semantic approach to visualizing online conversations. *Communications of the ACM*, 45(4), April 2002.

Judith Donath, Karrie Karahalios, and Fernanda Viegas. Visual who: Animating the affinities and activities of an electronic community. In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*. Computer Society Press, January 1999.

Judith Donath, Hyun-Yeul Lee, Danah Boyd, and Jonathan Goler. Loom2: Intuitively visualizing usenet. CSCW 2001 Workshop,
http://smg.media.mit.edu/papers/danah/CSCW2001PositionPaper.pdf, 2001.

Edd Dumbill. Finding friends with XML and RDF — The friend-of-a-friend vocabulary can make it easier to manage online communities.
http://www-106.ibm.com/developerworks/xml/library/x-foaf.html, June 2002.

Stephen G. Eick and Graham J. Wills. Navigating large networks with hierarchies. In *Proceedings of the IEEE Conference on Visualization*, Los Alamitos, CA, 1993.

C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):35–58, 1991.

Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erdmann, and Michael Klein. OIL in a nutshell. In Rose Dieng and Olivier Corby, editors, *Knowledge Acquisition, Modeling and Management*, pages 1–16. Springer, Berlin, 2000.

Dieter Fensel, Frank van Harmelen, and Ian Horrocks. OIL and DAML+OIL: Ontology languages for the semantic web. In John Davies, Dieter Fensel, and Frank van Harmelen, editors, *Towards the Semantic Web — Ontology-Driven Knowledge Management*, pages 11–31. John Wiley & Sons Ltd, Chichester, England, 2003.

David Ferraiolo and Richard Kuhn. Role-based access control. In *Proceedings of the 15th National Computer Security Conference*, 1992.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L.Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol — HTTP/1.1 (RFC 2616). http://www.faqs.org/rfcs/rfc2616.html, June 1999.

Danyel Fisher. Visualizing social newsgroup interaction. CHI 2000 Workshop: Electronic Communities: Places and Spaces, Contents and Boundaries, 2000.

Danyel Fisher and Christopher Lueg. Studying online newsgroups. In Lueg and Fisher (2003), pages 253–260.

Christiaan Fluit, Marta Sabou, and Frank van Harmelen. Ontology-based information visualization. In Vladimir Geroimenko and Chaomei Chen, editors, *Visualizing the Semantic Web – XML-based Internet and Information Visualization*, pages 36–48. Springer, London, Berlin, Heidelberg, New York, Barcelona, Hong Kong, Milan, Paris, Singapore, Tokyo, 2003.

Leonard N. Foner. What's an agent, anyway? A sociological case study. Agents Memo 93-01, Agents Group, MIT Media Lab, Cambridge, MA, 1993.

Leonard N. Foner. Yenta: A multi-agent, referral-based matchmaking system. In *Proceedings of the First International Conference on Autonomous Agents (Agents '97)*, 1997.

Eric J. Friedman and Paul Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2000.

Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software—-Practice and Experience*, 33(11):1203–1233, 2000.

Laura Garton, Caroline Haythornthwaite, and Barry Wellman. Studying online social networks. *Journal of Computer Mediated Communication*, 3(1), 1997. http://www.ascusc.org/jcmc/vol3/issue1/garton.html.

Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78:1360–1380, 1973.

Jonathan Grudin. Why cscw applications fail: Problems in the design and evaluation of organizational interfaces. Portland, Oregon, United States, 1988.

Jonathan Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1), January 1994.

R. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the twelfth international conference on World Wide Web*, pages 700–709. ACM Press, 2003.

Per Hage and Frank Harary. *Structural models in anthropology*. Cambridge Univ. Pr., Cambridge (and others), 1983.

Frank G. Halasz. The dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

Patrick Hayes, editor. *RDF Semantics*. 2003. W3C Working Draft 10 October 2003, http://www.w3.org/TR/2003/WD-rdf-mt-20031010/, work in progress.

Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1): 24–43, 2000.

P. Hoffman, L. Masinter, and J. Zawinski. The mailto URL scheme (RFC 2368). http://www.faqs.org/rfcs/rfc2368.html, July 1998.

Mark R. Horton and R. Adams. Standard for interchange of usenet messages (RFC 1036). http://www.faqs.org/rfcs/rfc1036.html, December 1987.

T. Howes and M. Smith. The LDAP URI format. http://www.faqs.org/rfcs/rfc2255.html, December 1997.

Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, 1990.

Ellen Isaacs, Alan Walendowski, Steve Whittaker, Diane J. Schiano, and Candace Kamm. The character, functions and styles of instant messaging in the workplace. In *Proceedings of the Conference on Computer-Supportet Cooperative Work (CSCW)*, New Orleans, LA, 2002.

Jupiterresearch. Total time spent using instant messaging jumps 110 percent at work and 48 percent at home versus last year, reports jupiter media matrix. Press Release, http://www.jupiterresearch.com/xp/jmm/press/2001/pr_111401.html, November 14, 2001.

Jupiterresearch. New instant messaging application poses threat to major services, according to media metrix internet ratings. Press Release, http://www.jupiterresearch.com/xp/jmm/press/2002/pr_052902.html, May 29, 2002.

Brian Kantor and Phil Lapsley. Network news transfer protocol: A proposed standard for the stream-based transmission of news (RFC 977). http://www.faqs.org/rfcs/rfc977.html, February 1986.

Henry Kautz, Bart Selman, and Mehul Shah. The hidden web. *AI Magazine*, 18(2):27–36, 1997a.

Henry Kautz, Bart Selman, and Mehul Shah. Referralweb: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997b.

Graham Klyne and Jeremy J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. 2003. W3C Working Draft 10 October 2003, http://www.w3.org/TR/2003/WD-rdf-concepts-20031010/, work in progress.

David Knoke. *Network analysis*. Sage Publ., Beverly Hills (and others), 1982.

Alfred Kobsa. Generic user modeling systems. In *User Modelung and User-Adapted Interaction 11*, pages 49–63. Kluver Academic Publishers, 2001.

Jürgen Hartmut Koch. *Unterstützung der Formierung und Analyse von virtuellen Communitites*. Peter Lang, Frankfurt am Main, Berlin, Bern, Bruxelles, New York, Oxford, Wien, 2003a.

Michael Koch. *Kooperation bei der Dokumentenbearbeitung — Entwicklung einer Gruppeneditorumgebung für das Internet*. Deutscher Universitäts-Verlag, Wiesbaden, 1997.

Michael Koch. An architecture for community support platforms — modularization and integration. In H. Luczak, A. E. Cakir, and G. Cakir, editors, *Proc. 6th Intl. Conf. on Work With Display Units — World Wide Work (WWDU2002)*, pages 533–535, 2002a.

Michael Koch. Global identity management to boost personalization. In Petra Schubert and Uwe Leimstoll, editors, *Proc. 9th Research Symp. on Emerging Electronic Markets*, pages 137–147, Basel, September 2002b.

Michael Koch. Community-Unterstützungssysteme — Architektur und Interoperabilität. Professorial dissertation, Department of Informatics, Technische Universität München, February 2003b.

Michael Koch, Michael Galla, and Helmut Schönenberger. Interoperable Community-Plattformen und Identitätsmanagement im Universitätsumfeld. In Martin Engelien and Jens Homann, editors, *Virtuelle Organisation und Neue Medien 2001 — Workshop GeNeMe2001 — Gemeinschaften in Neuen Medien*, pages 215–235. Josef Eul Verlag, September 2001.

Michael Koch, Kathrin Möslein, and Michael Wagner. Vertrauen und Reputation in Online-Anwendungen und virtuellen Gemeinschaften. In M. Engelien and D. Neumann, editors, *Virtuelle Organisation und Neue Medien*, pages 69–84. Josef Eul Verlag, Lohmar, Köln, 2000.

Michael Koch and Wolfgang Wörndl. Community-support and identity management. In *Proc. European Conference on Computer-Supported Cooperative Work (ECSCW2001)*, pages 319–338, Bonn, Germany, 2001.

Peter Kollock. The production of trust in online markets. In E. J. Lawler, M. Macy, S. Thyne, and H. A. Walker, editors, *Advances in Group Processes (Vol. 16)*. JAI Press, Greenwich, CT, 1999.

Wilfried Krüger. Konsequenzen der Globalisierung für Strategien, Fähigkeiten und Strukturen der Unternehmung. In Franz Giesel and Martin Glaum, editors, *Globalisierung — Herausforderung an die Unternehmensführung zu Beginn des 21. Jahrhunderts*. C. H. Beck, München, 1999.

John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. ACM Conf. Human Factors in Computing Systems, CHI*, pages 401–408. ACM, 1995.

Ora Lassila and Ralph R. Swick, editors. *Resource Description Framework (RDF) Model and Syntax Specification*. 1999. W3C Recommendation 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

Jonathan Lazar and Jennifer Preece. Classification schema for online communities. In *Association for Information Systems 1998 Americas Conference*, 1998.

Ulrike Lechner, Johannes Hummel, and Carl-Friedrich zu Inn- und Knyphausen. Peer-to-Peer Architekturen für Kollaboration in Communities. In Martin Engelien and Jens Homann, editors, *Virtuelle Organisation und Neue Medien 2001 — Workshop GeNeMe2001 — Gemeinschaften in Neuen Medien*, pages 237–253. Josef Eul Verlag, September 2001.

Vanda Lehel, Florian Matthes, and Klaus Steinfatt. Weblogs als ein innovatives Instrument des betrieblichen Wissensmanagements. In *Tagungsband der Konferenz Mensch & Computer 2003*, September 2003.

Jan Marco Leimeister, Miriam Daum, and Helmut Krcmar. Towoards m-communities: The case of cosmos healthcare. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS 36)*, 2003.

Amanda Lenhart, Lee Rainie, and Oliver Lewis. Teenage life online — the rise of the instant-message generation and the internet's impact on friendschips and family relationships. Pew Internet & American Life Project, http://www.pewinternet.org/reports/toc.asp?Report=36, 2001.

Liberty Alliance Project. Introduction to the liberty alliance identity architecture. White paper from http://www.projectliberty.org/resources/whitepapers/, 2003.

Greg Linden, Brend Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January/February 2003.

Christopher Lueg. Social filtering and social reality. In *Proceedings of the Delos Workshop on Collaborative Filtering*, Budapest, 1997.

Christopher Lueg and Danyel Fisher, editors. *From Usenet to CoWebs: interacting with social information spaces*. Springer, London, Berlin, Heidelberg, New York, Hong Kong, Mailand, Paris, Tokyo, 2003.

Alexander Maedche, Steffen Staab, Nenad Stojanovic, Rudi Studer, and York Sure. SEAL – A framework for developing SEmantic Web PortALs. *Lecture Notes in Computer Science*, 2097, 2001.

Eleni Mangina. Review of software products for multi-agent systems. Report (Draft), AgentLink II, the European Network of Excellence for Agent-Based Computing (IST-1999-29003), 2002.

M. S. Marshall, I. Herman, and G. Melançon. An object-oriented design for graph visualization. *Software — Practice and Experience*, 31(8):739–756, 2001.

Robert McArthur and Peter Bruza. Discovery of implicit and explicit connections between people using email utterance. In K. Kuutti, E. H. Karsten, G. Fitzpatrick, P. Dourish, and K. Schmitt, editors, *ESCW 2003: Proceedings of the Eighth European Conference on Computer Supported Cooperative Work, 14-18 September 2003, Helsinki, Finland*, pages 21–40, 2003.

Raúl Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 281–288. ACM Press, 1992.

Microsoft Corporation. Microsoft .NET Passport Review Guide. http://www.microsoft.com/net/services/passport/review_guide.asp (URL checked 2003), 2003.

Bradley N. Miller, John R. Riedl, and Joseph A. Konstan. Grouplens for usenet: Experiences in applying collaborative filtering to a social information system. In Lueg and Fisher (2003), pages 207–231.

George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11): 39–41, November 1995.

Nelson Minar and Judith Donath. Visualizing the crowds at a web site. In *CHI'99 late-breaking papers*. ACM Press, 1999.

Tamara Munzner. Exploring large graphs in 3d hyperbolic space. *IEEE Comput. Graph. Appl.*, 18(4): 18–23, 1998.

Elizabeth D. Mynatt, Annette Adler, Mizuko Ito, and Vicki L. O'Day. Design for nework communities. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press, 1997.

Bonnie A. Nardi, Steve Whittaker, and Erin Bradner. Iteraction and outeraction: Instant messaging in action. In *Proceedings of the 2000 ACM Conference on Computer supported cooperative work*, pages 79–88, New York, 2000. ACM Press.

Bonnie A. Nardi, Steve Whittaker, Ellen Isaacs, Mike Creech, Jeff Johnson, and John Hainsworth. Contactmap: Integrating communication and information through visualizing personal social networks. *Communications of the ACM*, 45(4):89–95, 2002.

H. Ogata, Y. Yano, N. Furugori, and Q. Jin. Computer supported social networking for augmenting cooperation. *Computer Supported Cooperative Work*, 10:189–209, 2001.

Mary O'Hara-Devereaux and Robert Johansen. *Global Work: Bridging Distance, Culture and Time*. Jossey-Bass, San Francisco, CA, 1994.

Open GIS Consortium, Inc. OpenGIS® geography markup language (GML) implementation specification, version 2.1.2. OpenGIS Project Document Number 02-069, http://www.opengis.net/gml/02-069/GML2-12.html (checked 2003), September 2002.

Osterman Research. Osterman research survey on instant messaging — highlights from a study conducted sepember 10–18, 2002.
http://www.ostermanresearch.com/results/surveyresults_im0902.htm, 2002.

Jacob Palme. Common internet message headers (RFC 2076). http://www.faqs.org/rfcs/rfc2076.html, February 1997.

Uta Pankoke-Babatz, editor. *Computer Based Group Communication. The AMIGO Activity Model*. Ellis Horwood Limited, 1989.

H. Van Dyke Parunak and James Odell. Representing social structures in UML. In *Proceedings of the Workshop on Agent-Oriented Software Engineering (AOSE) 2001 (at Autonomous Agents 2001 Conference)*, 2001.

Peter F. Patel-Schneider and Dieter Fensel. Layering the semantic web: Problems and directions. In *The Semantic Web — ISWC 2002*, pages 16–29, June 2002.

Bryan Pfaffenberger. "A standing wave in the web of our communications": Usenet and the socio-technical construction of cyberspace values. In Lueg and Fisher (2003), pages 20–43.

Jeanne M. Pickering and John Leslie King. Hardwiring weak ties: Individual and institutional issues in computer mediated communication. In *Proceedings of the 1992 ACM conference on Computer Supported Cooperative Work*, pages 356–361, Toronto, Ontario, Canada, 1992.

Arnold Picot, Ralf Reichwald, and Rolf Wigand. *Die Grenzenlose Unternehmung*. Gabler, Wiesbaden, 1996.

Wolfgang Prinz. Survey of group communication models and systems. In Pankoke-Babatz (1989), pages 127–180.

RDFWeb.org. The 'friend of a friend' project: FOAF. Website (deployed but changing), http://rdfweb.org/foaf/, 2003.

Ralf Reichwald and Kathrin Möslein. Auf dem Weg zur virtuellen Organisation: Wie Telekooperation Unternehmen verändert. In Ralf E. Strauß Günter Müller, Ulrich Kohl, editor, *Zukunftsperspektiven der digitalen Vernetzung*, pages 209–233. dpunkt Verlag, Heidelberg, 1996.

Ralf Reichwald and Kathrin Möslein. The virtual company. In *European conference on the future of working conditions*. BAuA Publications, June 8–9 1999.

Ralf Reichwald, Kathrin Möslein, Hans Sachenbacher, and Hermann Englberger. *Telekooperation — Verteilte Arbeits- und Organisationsformen*. Springer, London, Berlin, Heidelberg, New York, Hong Kong, Mailand, Paris, Tokyo, second edition, 2000.

Paul Resnick and Hal R. Varian. Recommender systems. *Introduction to special section of Communications of the ACM*, 40(3), March 1997.

Howard Rheingold. *The Virtual Community. Homesteading on the Electronic Frontier.* Addison-Wesley, Reading (Massachusetts), Menlo Park (California), New York, Don Mills (Ontario), Wokingham (England), Amsterdam, Bonn, Sydney, Singapore, Tokyo, Madrid, San Juan, Paris, Seoul, Milan, Mexico City, Taipei, 1993.

Martin Röscheisen and Terry Winograd. A network-centric design for relationship-based security and access control. *Journal of Computer Security, Special Issue on Security in the World-Wide Web*, 1997.

R. Martin Röscheisen. *A Network-Centric Design for Relationship-Based Rights Management.* PhD thesis, Stanford University, December 1997.

Warren Sack. Conversationmap: A content-based usenet newsgroup browser. In Lueg and Fisher (2003), pages 92–109.

Johann Schlichter, André Büssing, Ralf Reichwald, Michael Galla, Claudia Moranz, and Michael Wagner. Wissen und Vertrauen bei der Kontaktanbahnung in Gründernetzwerken. In Peter Mambrey, Volkmar Pipek, and Markus Rohde, editors, *Wissen und Lernen in virtuellen Organisationen*. Physica-Verlag, Heidelberg, 2003.

Johann Schlichter, Michael Koch, and Chengmao Xu. Awareness — the common link between groupware and communityware. In Toru Ishida, editor, *Community Computing and Support Systems*, pages 77–93. Springer, Heidelberg, 1998.

Marc Smith. Invisible crowds in cyberspace: Measuring and mapping the social structure of usenet. In Marc Smith and Peter Kollock, editors, *Communities in Cyberspace*. Routledge Presse, London, 1999.

Marc A. Smith. Measures and maps of usenet. In Lueg and Fisher (2003), pages 47–78.

Marc A. Smith and Andrew T. Fiore. Visualization components for persistent conversations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 136–143, Seattle, Washington, United States, 2001.

Michael K. Smith, Chris Welty, and Deborah McGuinness, editors. *OWL Web Ontology Language Guide*. 2003. W3C Working Draft 31 March 2003, http://www.w3.org/TR/2003/WD-owl-guide-20030331/, work in progress.

Henry Spencer. News article format and transmission. "Son-of-1036" (http://www.chemie.fu-berlin.de/outerspace/netnews/son-of-1036.html), June 1994.

Steffen Staab, Michael Erdmann, and Alexander Maedche. Ontologies in RDF(S). *Linköping Electronic Articles in Computer and Information Science*, Vol. 6(2001): nr 9, December 2001. http://www.ep.liu.se/ea/cis/2001/009/.

Stephanie Teufel, Christian Sauter, Thomas Mühlherr, and Kurt Bauknecht. *Computerunterstützung für die Gruppenarbeit*. Addison-Wesley, Bonn, 1995.

Peter D. Turney. Extraction of keyphrases from text: evaluation of four algorithms. Technical Report NRC 41550, National Research Council of Canada, 1997.

Peter D. Turney. Coherent keyphrase extraction via web mining. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 434–439, Acapulco, Mexico, 2003.

Fernanda Viegas. Post history. http://web.media.mit.edu/∼fviegas/posthistory/, 2002.

Adriana S. Vivacqua. Agents for expertise location. In *Proceedings of AAAI Spring Symposium on Intelligent Agents in Cyberspace*, Stanford, CA, United States, March 1999.

Samuel Warren and Louis D. Brandeis. The right to privacy. *Harvard Law Review*, 4(5), 1890.

Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

Gerhard Weiss, editor. *Multiagent Systems — A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts, 1999.

Barry Wellman. For a social network analysis of computer networks. In *Proceedings of the 1996 conference on ACM SIGCPR/SIGMIS conference*, pages 1–11, Denver, Colorado, United States, 1996.

Barry Wellman. An electronic group is virtually a social network. In Sara Kiesler, editor, *Cultures of the Internet*, pages 179–205. Lawrence Erlbaum Publications, Mahwah, New Jersey, 1997.

Barry Wellman. Computer networks as social networks. *Science*, 293, September 2001.

Barry Wellman. Designing the internet for a networked society: Little boxes, glocalization, and networked individualism. *Communications of the ACM*, 45(5), May 2002.

Barry Wellman, Jeffrey Boase, and Wenhong Chen. The networked nature of community: Online and offline. *IT&Society*, 1(1):151–165, 2002.

Barry Wellman and Milena Gulia. Net surfers don't ride alone: Virtual communities as communities. In *Communities in Cyberspace*, pages 167–194. Routledge, London, 1999.

Barry Wellman and Keith Hampton. Living networked in a wired world. *Contemporary Sociology*, 28(6), 1999.

Etienne C. Wenger and William M. Snyder. Communities of practice: The organizational frontier. *Harvard Business Review*, January/February 2000.

Steve Whittaker, Ellen Isaacs, and Vicky O'Day. Widening the net: Workshop report on the theory and practice of physical and network communities. *SIGCHI Bulletin*, 29(3), 1997.

Terry Winograd. A language/action perspective on the design of cooperative work. *Human-Computer Interaction*, 3(1):3–30, 1988.

Marianne Winslett, Ting Yu, Kent E. Seamons, Adam Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, 2002.

Michael Wooldridge. Intelligent agents. In Weiss (1999), pages 27–77.

Wolfgang Wörndl. Privatheit bei dezentraler Verwaltung von Benutzerprofilen. Doctoral thesis, Department of Informatics, Technische Universität München, August 2003.

Hans A. Wüthrich, Andreas F. Philipp, and Martin H. Frentz. *Vorsprung durch Virtualisierung — Lernen von virtuellen Pilotunternehmen*. Gabler, Wiesbaden, 1997.

Rebecca Xiong and Eric Brittain. Liveweb: Visualizing live user activities on the web. Siggraph 1999 conference, Sketches & Applications, 1999.

Song Yang and David Knoke. Optimal connections: Strength and distance in valued graphs. *Social Networks*, 23(4):285–295, 2001.

Index