Patterns of Virtual Collaboration

Robert P. Biuk-Aghai

A thesis submitted for the degree of
Doctor of Philosophy in Computing Sciences
at the
University of Technology, Sydney

Faculty of Information Technology
University of Technology, Sydney

Sydney, Australia

2003

To my parents, with love and gratitude

Certificate of Authorship/Originality

Abstract

Virtual collaboration—the act of working together across boundaries of space, time, and organization, aided by technology—has become increasingly commonplace in recent years. Doing so, however, presents a number of challenges to those involved. One of these is that because of a lack of experience in collaborating through computer-based collaboration systems, there is little knowledge on how to carry out collaboration virtually. Another is that it is not easy for those not directly involved in the collaboration to know what is, and has been, "going on" during virtual collaboration. This thesis suggests that both of these challenges can be addressed with the same approach, namely by referring to observations of virtual collaboration. The problem then is how such observations of virtual collaboration can be obtained without requiring those involved in it to document their own actions. To address this problem is the objective of this thesis.

The approach proposed here involves three elements: firstly, the collection of data about virtual collaboration; secondly, the modeling of this data; and thirdly, the derivation of increasingly abstract, larger-scale representations of virtual collaboration from this data. These representations are termed *patterns of virtual collaboration*, which are abstract descriptions of activities of virtual collaboration. A multi-layered conceptual model of information, the *Information Pyramid of Virtual Collaboration*, is proposed, providing different views of information related to virtual collaboration, at different levels of abstraction. The thesis then suggests how from a given body of data, patterns of virtual collaboration at a corresponding level of the Information Pyramid can be extracted, and how from collections of such patterns more abstract patterns of larger-scale activity can be derived, providing the observations of virtual collaboration sought.

In considering how the extraction of patterns of virtual collaboration fits into the larger context of the conception, design, and use of collaboration systems, a *Framework for Pattern Extraction and Feedback* is proposed. This framework introduces the notion of *collaboration memory*, a type of organizational memory that contains records of collaborative activity. Moreover, the framework suggests how extracted patterns of virtual collaboration feed back into both ongoing development and use of collaboration systems.

Finally, the modeling and extraction of patterns of virtual collaboration is illustrated in a case study involving the LIVENET collaboration system.

Acknowledgments

At the end of a long and arduous undertaking such as doctoral research, great thanks are due to many people who have been of help and support along the way:

First and foremost to my supervisor Igor Hawryszkiewycz, who by email accepted me as a PhD student without ever having met me, and let me spend my first year as a PhD student overseas (still without having met me!). Thus, at least in part, my PhD work was carried out in a form of virtual collaboration, the very thing which this research is about. Igor not only gave me much guidance, but also a great degree of freedom, for both of which I am very grateful.

To my co-supervisor John Debenham, who always was ready to discuss my research (on the few occasions that I ventured into his office), and who never failed to inundate me with a torrent of ideas of what I could or should consider, as well as numerous comments on various drafts of thesis chapters. My thesis is so much the better because of this.

Great thanks are due to Simeon Simoff, a good colleague and friend whom I had the great fortune of collaborating with. In particular, the development of the ideas presented in Section 4.5 are based on earlier joint work performed by Simeon and myself (Biuk-Aghai and Simoff, 2001). Looking forward to many more fruitful collaborations!

To my office colleagues Ingrid Slembek, Alan Lin, and Dongbai Xue whom I hassled more than once, and who were always willing to help and listen.

To the colleagues from the School of Management, Thekla Rura-Polley and Ellen Baker. We have overlapping research interests, and it was great and enlightening to work with them.

To the technical and admin staff at the Faculty of Information Technology, for their usually prompt and helpful support, particularly: Malik, Alex, Graham, Bj, and sbg.

To a number of colleagues in the faculty who made comments along the way which no doubt got reflected in the thesis one way or another: Maolin Huang, Valérie Gay, Richard Raban, and Toni Robertson.

To the DSTC staff with whom I had fruitful discussions, or who simply were helpful when help was needed: Luke Cole, Kim Dinh, Glen “Marty” MacLarty, and Tim Mansfield.

To my employer, the University of Macau, for their willingness to invest in my future and maintain my position for me while I was overseas for two years.

To the makers of those wonderful software tools which I could not have done without in preparing this thesis, mainly LaTeX, BibTeX, and XFig.

Not least of all to my wife Ouling and my daughter Chuer for their continuing love and support, and for having endured five years of my long nights and weekends of work, as well as several months absence when I was overseas for conferences or work.

Finally, to my parents who gave me the gift of life and enabled me to get where I am now. I am deeply grateful to them.

Contents

List of Figures

List of Tables

List of Definitions

Chapter 1

Introduction

This thesis is concerned with obtaining insights into the work of virtual teams who collaborate through the support of computer-based systems. Such systems can collect large amounts of fine-grained data about events transpiring in them (e.g. user logs in, user opens a discussion forum, user uploads a document, etc.). This thesis suggests how collections of such fine-grained events can be aggregated to more abstract representations of work activities such as tasks and processes (e.g. reviewing a paper). It also suggests how these abstract representations can contribute in an ongoing manner to an organization's records of its own actions. The following sections outline the issues that motivate this research and the problems it attempts to address.

1.1 The Changing Organizational Setting

The following discussion is concerned with organizations and the changes experienced by them. In this context, the term "organization" is understood to refer to any entity that forms a social structure comprised of members, be they individuals or in turn other organizations. For the most part, the comments made in this section apply to business organizations specifically, but many of the issues raised also bear upon other types of organizations, such as governmental, educational, and other organizations.

In past decades, many business organizations could operate in relatively stable and predictable environments:

> By the late 1970s, the industrial economy had been chugging along for almost a century, and, for the most part, its structure was fixed and competition was predictable. (Carr, 2001, p. ix)

With the introduction and spread of information and communication technology, however, this relative stability has started eroding, and this trend has been particularly marked and accelerated in the past decade: "In recent years, it seems as though the only constant in business has been upheaval" (Sawhney and Parikh, 2001). Greater amounts of more easily accessible information, and more open, increasingly global markets, have influenced the way organizations are run and business is conducted. Business organizations are operating in increasingly competitive environments, with some sectors of the economy operating under such great competitive pressures that the management community has coined a new term to describe this phenomenon: *hypercompetition* (D'Aveni, 1994; Naff, 1995). This greatly increased competition has given rise to an equally greatly increased pace of operation. For instance, products and services are no longer conceived, developed and launched in timeframes that are measured in years, but for many sectors of the economy this cycle time has been reduced to months or less:

> New products, even whole markets, appear, mutate, and disappear within shorter and shorter periods of time. The pace of innovation continues to quicken, and the direction of innovation is often unpredictable. (Goldman et al., 1995, p. 3)

The impact this has had, and is having, on the affected organizations is profound:

> Changes have occurred at every level, from the way entire industries are structured, to the way companies interact with customers, to the way basic tasks are carried out in individual organizations. (Sawhney and Parikh, 2001)

1.2 Virtual Teams and Virtual Collaboration

Coping with these pressures and ensuring survival of the organization often requires great agility, i.e. the ability of the organization to quickly and nimbly reorganize itself, in part or in whole, in response to a given stimulus. The concept of the *agile company* has been proposed as a solution to the changing organizational setting (Goldman et al., 1995). This type of organization often employs new organizational forms (Coleman and Khanna, 1995, p. 3) such as cross-functional teams (i.e. teams whose members come from different functional areas, and usually also from different organizational units) (Goldman et al., 1995; Richards and Makatsoris, 2002) and virtual teams. While a conventional team is understood to be "a group of people organized to work together"[1], a virtual team has been defined as follows:

[1]The Collins Concise Dictionary of the English Language, 2nd edition, 1988.

Definition 1 *A **virtual team**, like every team, is a group of people who interact through interdependent tasks guided by a common purpose. Unlike conventional teams, a virtual team works across space, time, and organizational boundaries with links strengthened by webs of communication technologies (Lipnack and Stamps, 1997).*
□

A virtual team may be assembled only for the duration of a given project, cutting across the organizational hierarchy and integrating otherwise separate human resources. When these resources are physically distributed, the use of information and communication technology facilitates collaboration of these teams, despite their members' physical distance (Steinfield, 2002). It has been suggested that the organization of the 21st century will be made up of virtual teams and of networks of such teams (Lipnack and Stamps, 1999).

Members of virtual teams interact with each other in a mode of *collaboration* or *cooperation*. These two terms are defined here as follows:

Definition 2 ***Collaboration** is the act of working together on a common task or process.*
□

Definition 3 ***Cooperation** is the joint operation or action toward a common goal or benefit.*
□

The meaning of these two terms seems to be almost the same, and indeed in the literature they are often used interchangeably. The connotations associated with each, however, are slightly different and are best highlighted by considering their antonyms: the antonym of collaboration is "working independently", while that of cooperation is "competition"—an entirely different notion. Therefore, when talking about the work of virtual teams in the context of this thesis, the term "collaboration" is used instead of "cooperation", to emphasize the fact of working together.

Virtual collaboration, then, is defined as follows:

Definition 4 ***Virtual collaboration** is collaboration which is conducted without face-to-face interaction, enabled by technology.*
□

		Time	
		same	*different*
Place	*same*	Same Time Same Place	Different Time Same Place
	different	Same Time Different Place	Different Time Different Place

Figure 1.1: Time-place matrix of collaboration, adapted from (Johansen et al., 1991)

Most commonly this technology is a computer-based system, but virtual collaboration can also make use of other tools such as telephone, fax, video-conferencing systems, etc.

Virtuality of teams and collaboration is not an absolute measure, but rather is a continuum that ranges between the completely non-virtual and the completely virtual. An early classification of collaboration support systems was proposed in (Johansen et al., 1991) based on the two dimensions of *time* and *place* of the collaboration, and their distribution among the two poles of *same* and *different*, yielding the now well-known matrix shown in Figure 1.1. Virtual collaboration as defined here thus matches all quadrants except the top-left one (same time, same place).

However, this classification is somewhat simplistic in its assumption that collaborative work fits neatly into one of the four areas of the matrix, i.e. that the totality of collaborative activity can be confined within that quadrant. In reality, it is more common for a mix of these different conditions to exist. For instance, for a joint authoring task an initial face-to-face meeting (same time, same place) may be followed by a period of individual writing activity (different time, different place), followed by a concluding period of integration of the separate sections of the joint work aided by collaboration technology (same time, different place).

Other classifications have been put forward, such as the one proposed in (Niederman and Beise, 1999), which is concerned with defining virtuality of groups, teams, and meetings. It suggests that rather than classifying by the dimensions of time and place, the preferred dimensions for classification should be the extent to which technology is used, and to which technology use is combined with face-to-face interactions. Values in each of these two dimensions range from *low* to *high*. The resulting classification takes the form shown in Figure 1.2. In this framework, the *highly-virtual* category, where face-to-face interactions are low and electronic mediation is high, most closely matches the definition of virtual collaboration adopted here.

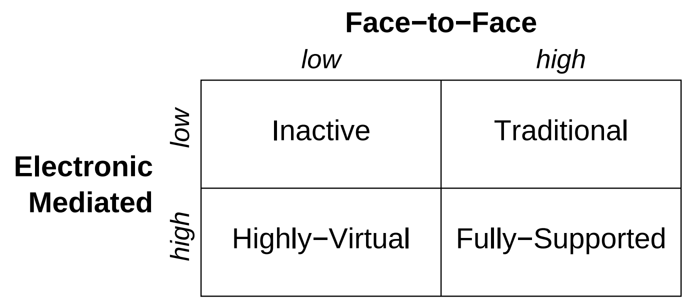


Figure 1.2: Categories of virtual groups, teams, and meetings, from (Niederman and Beise, 1999)

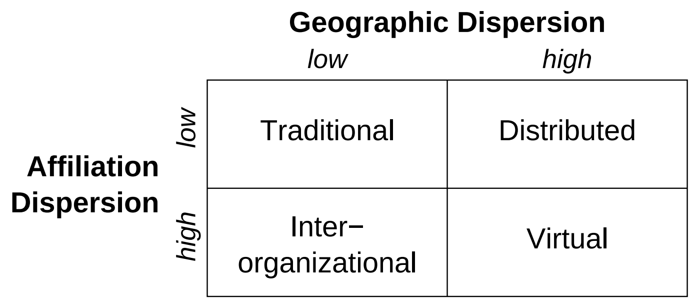


Figure 1.3: Categories of virtual projects, from (Katzy et al., 2000)

Virtual teams may also span organizational boundaries, where separate organizations that otherwise compete with each other decide to pool their resources in order to jointly achieve what each one of them singly is not able to. Indeed, entire organizations may decide to form temporary alliances that present themselves to the outside as a single organization, so-called *virtual organizations* (Donlon, 1997)[2].

Yet another classification of virtuality, in the context of *virtual projects*, that includes organizational boundary-spanning virtual teams has been proposed in (Katzy et al., 2000) and is shown in Figure 1.3. This classification includes the dimension of *affiliation dispersion*, which expresses the extent to which members of a project team belong to the same or different organizations. The other dimension is *geographic dispersion*, which is the same as the *place* dimension in the classification of (Johansen et al., 1991). Values in each dimension range from *low* to *high*. As this classification is specifically concerned with the aspect of affiliation dispersion, it does not map clearly into the definition of virtual collaboration adopted here. In all quadrants it is possible for virtual collaboration to

[2]An example of the application of the concept of virtual organizations can be found in movie production, where a large number of independent companies and individuals come together for the duration of the project, then disband.

take place, although in the two left ones (low geographic dispersion) the collaboration is more likely to be of the face-to-face type, rather than being virtual.

The trend towards virtual teams and virtual organizations leads to an increased blurring of both inter- and intra-organizational boundaries, along with an increased degree of virtual collaboration. As a result, a greater portion of an individual's activity shifts from the physical to the virtual world. This offers a number of opportunities, among which are cost and time savings, greater accessibility of team members regardless of time zone or location, ability to participate in multiple projects at the same time, etc. However, this shift also presents a number of challenges, some of which are discussed in the following section.

1.3 Challenges

People have centuries of experience working in collaboration with each other. Until very recently, however, this collaboration has practically always been of the face-to-face kind, in more or less close physical proximity. Not until the advent of modern telecommunication technology has it become possible to effectively carry out collaboration in any other way[3]. Consequently, appropriate methods and techniques for communication, collaboration, management and coordination in the physical world were developed over a long period of time.

For instance, if a group of people is to work closely together, experience advises that it is best to bring them together in close physical proximity to each other; to ensure that they come together at the same time; to make tools and materials required for carrying out the work easily accessible within the working environment; to place the most frequently used of these closest to those who make the most use of them; etc. It is known that the physical arrangement of different members of a collaborative work group should take features of the work process into account. It is known that this arrangement is also dependent on the nature of the work carried out: sometimes it is better for people to share an open office where they are afforded the opportunity to see and hear each other, while at other times it is better for them to have their own separate offices, such as when their work requires a quiet environment to support periods of concentrated thinking. Based on the long experience of working together in the physical world, these kinds of insights come

[3]Means of communication other than through face-to-face interactions did exist in past centuries, including the use of carrier pigeons, drum telegraphs, light and smoke signals, couriers, and later the mail service and the electric telegraph. However, it can be argued that the limitations of bandwidth and/or latency of these former means of communication made the kind of collaboration considered here all but impossible.

naturally and do not require much, or indeed any, deliberation.

In the virtual realm, however, the situation is very different. Because collaboration conducted through the aid of information and communication technology is a very recent phenomenon, there is very little experience in how to carry out collaboration virtually. Therefore it is difficult to know how a virtual work environment should be designed, and how the work should be structured. With what items should the work environment be furnished? Should it be structured in a similar manner as one does with a physical work environment? Should one perform all tasks in the same environment or in different ones? What links should exist between different environments? How should distinct tasks be related with one another? It is tempting to simply apply one's experience of face-to-face collaboration to virtual collaboration, but whether or not these experiences are in fact transferable to the virtual realm in that form is not certain. Thus there is the following challenge:

Challenge 1 *How can one know how to carry out collaboration virtually?*

Moreover, one of the difficulties associated with the virtual world is its lack of many of the affordances which the physical world provides. In the physical world, the traditional sensory modes of perception, primarily visual and auditory, provide a continuous rich source of information on events transpiring in the working environment, both the immediate and the less immediate one. Thus, for instance, when a nearby co-worker performs an action, one can immediately become aware of it, which may then influence one's own work. On the other hand, in the virtual realm one's view of the world is very much impoverished, reduced to several square decimeters of screen space, which at best may be supplemented by an audio channel. Thus knowing what is going on in the virtual work environment is encumbered by the limitations of the technology. Researchers in the HCI (human-computer interaction) and CSCW (computer-supported cooperative work) communities have long studied this problem of *awareness*. Much of the work in this area has focused on delivering fine-grained events to those present in a virtual work environment, often with an emphasis on synchronous (same-time) collaboration. This can be of value to those directly engaged in highly inter-connected work, such as for example in collaborative editing. However, to those less directly involved in specific work tasks, including other colleagues as well as management, such detailed awareness information often is of little value, and may in fact contribute to *information overload* (Hiltz and Turoff, 1985; Fussell et al., 1998). Instead, what is required rather than detailed up-to-the-second event traces is a high-level overview of the tasks performed by both individual team members and entire teams, as well as their progress over time, including both the present, as well as the history of actions that have already transpired. Thus another challenge posed is this:

> **Challenge 2** *How can one know what is, and has been, "going on" during virtual collaboration?*

The history of a given virtual team's collaboration is particularly relevant to those who have not been immediately involved in it—new team members who need to get "up to speed" on what the team has been doing prior to their joining it, or management who are interested in the work performed by a team. The history of an organization's actions is one of the areas of research into *organizational memory*, which for over a decade has been addressing the problem of organizations suffering the equivalent of *amnesia*, i.e. the inability to recall their past actions and bring that knowledge to bear on present endeavours. Virtual collaboration exacerbates the problem of organizational amnesia because, taking place in virtual space, it is less directly observable than activity in physical space.

The following section suggests how these challenges can be met.

1.4 Research Problem

The two challenges described above, that is, knowing how to carry out collaboration virtually, and knowing what has transpired during virtual collaboration, form the motivation for this research. Here it is suggested that meeting these two challenges can be addressed with the same approach.

Knowing how to carry out collaboration virtually requires knowing how to design the virtual work environment, and knowing in what way to utilize it to perform specific tasks. Drawing the parallel with the physical world: how does one know what resources are required for a task in a physical work environment, how they should be related, and how they should be utilized for the performance of the task? Unless one has already performed that task before, there are a number of possibilities: for instance, one may study work manuals that describe required resources and methods of execution; alternatively one may resort to trial-and-error; one could ask someone whom one knows to be experienced; or one could observe someone else perform the task. It is argued here that in most cases observation is preferable to the other approaches: observation is practical, in that one can actually perceive the task in action, rather than merely a description of it; and this observation allows one to gain from the experience of the one performing the task, rather than having to attempt to gain this experience by trial-and-error.

Thus the answer to the first challenge, "how can one know how to carry out collaboration virtually?" is this: by observing the virtual collaboration of others.

The answer to the second challenge, "how can one know what is, and has been, "going on" during virtual collaboration?", on the other hand, can be answered in the same way: by observing the virtual collaboration. Thus meeting the two challenges can be addressed with the same approach.

However, this raises a new challenge, namely how virtual collaboration can be observed. This is problematic for a number of reasons. Firstly, collaboration in the virtual world is not direct but always needs to be *mediated*. For instance, communication is mediated through some form of communication channel which relays "utterances" between the communicating parties; pointing at an object is not performed directly but is mediated through, for example, some form of graphical pointer; handing a document to another person too is not performed directly but is mediated through some form of electronic document exchange mechanism. Thus collaboration consists of a number of separate mediated actions, which have to be pieced together in order to perceive the collaboration performed.

A second problem is that the time and location of observers and the collaboration being observed may be significantly different. Timezone differences of more than a few hours make it impractical to directly observe collaboration as it takes place. Even at the same time, observation requires observers to focus on the same location within the virtual work environment as where the collaboration of that moment takes place. However, unlike in a physical work environment where the movement and actions of people within the environment is easily perceived, in the virtual environment this can be much more difficult to perceive.

The greatest problem with direct observation, however, is when the history of a collaboration needs to be revisited. In this case, direct observation, in the sense of observing something as it takes place, is not possible as the object of observation has already transpired.

It is argued here that instead it is necessary to maintain a record of a virtual team's collaboration. However, requiring teams to explicitly document their own actions is also problematic. Firstly, doing so would interfere with the collaboration. Secondly, unless a large enough benefit for doing so could be perceived on the part of the team, there would be no incentive to maintain such records, and consequently it is questionable whether accurate and complete records would be maintained. Finally, records from different teams and different individuals would necessarily be subjective, which would make comparison difficult.

Thus the main problem to be addressed in this research is this:

> **Problem:** *How can information on the collaborative activities/practices of virtual teams be obtained without requiring these teams to explicitly document their own actions?*

The methodology and approach for addressing this problem are outlined next.

1.5 Research Methodology and Approach

1.5.1 Research Methodology

This research explores the means for understanding virtual collaboration through the following main steps:

1. First, the *argument* for the need for obtaining descriptions of virtual collaboration is made.
2. Next, the problem domain is surveyed in order to lay out the *issues of relevance* to the problem.
3. This is followed by the definition of the main *concepts* and approach proposed in this research for addressing the problem.
4. Based on these developed concepts, *methods* are developed to enable these concepts to be applied to the problem.
5. Finally, the methods are *applied* to a case study to demonstrate their practical applicability.

This research mainly sets out to demonstrate the *plausibility* of the proposed concepts and methods in addressing the research problem. It includes limited testing of the concepts and methods, leaving rigorous empirical testing in a wide range of real-world situations for future research. Here it is argued that this approach is justified, as there are currently no concepts and methods for systematically addressing the research problem. The focus of the thesis is therefore on the development of concepts and methods. Validation of the developed concepts and methods is, however, obtained to some extent through their application to a large case study. The methodology followed here is based on that of earlier related work published in (de Moor, 1999).

1.5.2 Research Approach

This research proposes that high-level descriptions of virtual collaboration can be produced without the involvement of those performing the collaboration. In outline, this involves the following:

1. Availability of *records of events* transpiring during virtual collaboration, capturing information on content and context of these events. Such records should be automatically collected by the computer-based system through which virtual collaboration is carried out, without human involvement.

2. *Conceptual modeling* of information of the computer-based systems through which virtual collaboration is carried out. The conceptual model should have the expressiveness to represent collected records of events.

3. *Derivation* of successively more abstract, large-scale representations of virtual collaboration. Given a conceptual model and collected records of events represented in terms of the conceptual model, mappings to more large-scale units of information can be defined. Records of events can then be mapped to successively more abstract levels until records represent entire work tasks and processes.

Each of these items is elaborated in later chapters. Key elements of this approach have been implemented in an existing computer-based system for virtual collaboration, LiveNet, which was developed at the University of Technology, Sydney's Collaborative Systems Laboratory. Relevant sections of the thesis therefore illustrate practical issues on this system.

1.6 Outline of the Thesis

In outline, the remainder of the thesis is structured as follows:

Chapter 2 reviews the main components of the problem domain: virtual collaboration processes; collaboration systems, through which virtual collaboration processes are performed; and organizational memory, which can maintain a history of virtual collaboration.

Chapter 3 then develops principles for the modeling of information of collaboration systems. It proposes a multi-layered conceptual model for the representation of information about collaboration at different levels of abstraction, and suggests how concepts at each level can be represented in the form of patterns of virtual collaboration.

Chapter 4 carries on from there to propose details of the specification and extraction of patterns of virtual collaboration, elaborating on several of the pertinent issues involved. It then goes on to consider the larger context of pattern extraction by proposing an integrating framework that suggests how extracted patterns can feed back into the virtual collaboration.

Chapter 5 applies the concepts and methods of the preceding two chapters in a case study of the extraction of patterns of virtual collaboration from data collected by the LiveNet collaboration system.

Chapter 6 concludes this thesis by giving a summary of the points presented, contributions made, opportunities for future work, and by offering some concluding remarks.

1.7 Typographic Conventions

In order to more easily distinguish symbols with special meaning in the text, a number of different typefaces are used:

- The names of concepts within an ontology are typeset in a sans-serif font (e.g. Discussion-Forum).
- The names of classes and slots within an ontology, and ontology specifications and functions are typeset in a courier font (e.g. `LN-Discussion-Forum`).
- Names of collaboration systems are typeset in small caps (e.g. LiveNet), except where the name is an abbreviation already consisting of all capitals (e.g. BSCW).

Chapter 2

The Problem Domain

As the previous chapter has introduced, the research presented here is concerned with virtual collaboration, and with maintaining observations of such collaboration. In the present chapter, an overview of the three core areas which make up the problem domain is given: (1) virtual collaboration processes, (2) collaboration systems, and (3) organizational memory. This overview provides the basis for the contributions of later chapters.

2.1 Virtual Collaboration Processes

The introduction briefly discussed the trend within many organizations towards virtual teams and virtual collaboration. Here, this discussion is continued by considering the virtual collaboration processes carried out by virtual teams. First, the kinds of collaboration processes that can be distinguished are reviewed. This is followed by an overview of some of the notations available for representing such processes.

2.1.1 Kinds of Virtual Collaboration Processes

When considering work processes in general, many kinds of classifications are possible. For instance, work processes can be classified by broad category, such as manual vs. intellectual work; by detailed category, such as product design, marketing, etc; by temporal features, such as ongoing vs. intermittent; by spatial features, such as fixed vs. mobile; and so on. However, since the focus of this thesis is on *collaborative* work, the present discussion is concerned with collaboration processes specifically, rather than its opposites: competitive processes (more than one person working in competition), and independent processes (one person working alone). More specifically, this collaboration is understood to be virtual collaboration, as opposed to face-to-face collaboration.

2.1.1.1 Definitions

To repeat from Chapter 1, *collaboration* is understood to be "the act of working together on a common task or process". This act of working together involves specific *activities*, performed by individuals. Examples of activities are: reading a report, or commenting on a paper. An activity may (but does not have to) be performed in order to bring about a specific *goal*. A goal, in turn, refers to the desired realization of a specific state of a portion of the world that the activity is concerned with. So, for the above example of commenting on a paper, the goal could be to come up with a list of changes needed for getting the paper ready for publication. The goal may not always be achieved, however, and so the outcome of an activity may be different from its goal. Outcome refers to the actual (as opposed to the desired) realization of a specific state of the portion of the world affected by the task.

A collection of several activities may have a common goal, where each activity contributes to the achievement of the goal. Such collections of activities with common goals are referred to as *tasks*. Following on from the above example, the task which the mentioned activity is part of could be that of reviewing a paper. This task could involve other activities, such as reading the paper, and writing up a list of required amendments to the paper. Each of these activities contributes to the achievement of the task's goal, which in this case could be that of completion of a paper review. Thus a definition of "task" is as follows:

> **Definition 5** *A **task** is a collection of activities with a common goal, performed by one or more individuals, such that the successful completion of all the activities brings about the task's goal.*
> □

A process is a collection of related tasks. Processes also have goals; the accomplishment of the goals of all the tasks belonging to a process brings about the process goal. Unlike tasks and activities, where the task's activities have a common goal, tasks in a process do not have a common goal, and their goal typically is not identical to that of the process which they are part of. However, task goals are subsumed under, and contribute to, the goal of the process they are part of. An example of a process is that of producing the proceedings of a conference. It involves many tasks, such as recruiting paper reviewers, calling for paper submissions, reviewing papers, etc. Each of these tasks has its own goal, but all contribute to the goal of the entire process, which is the publication of the conference proceedings. A definition of "process" follows:

Definition 6 *A **process** is a collection of related tasks with a goal, such that the accomplishment of all task goals brings about the process goal.*
□

Activities, tasks and processes thus all are units of human activity, however at different levels of granularity. Task and activity, as well as process and task are related through a whole-part relationship. Between the parts, i.e. between the activities of a task or between the tasks of a process, there are typically also relationships, such as dependencies; for example, there may exist a dependency that one task can only start when another task has been completed. Finally, the definition of process provided here purposely considers it to be made up of a *collection* of tasks, rather than a series or sequence. Thus, a process may well consist of temporally overlapping or parallel tasks. The same comments apply for the activities of a task.

A *collaboration process*, then, is defined as follows:

Definition 7 *A **collaboration process** is a process performed by two or more individuals working together.*
□

Finally, bringing this definition together with that of virtual collaboration given in the previous chapter, a *virtual collaboration process* is defined as follows:

Definition 8 *A **virtual collaboration process** is a collaboration process performed without face-to-face interaction, enabled by technology.*
□

Different kinds of virtual collaboration processes may be distinguished, and can be classified according to a number of differentiating attributes. Here the following set of attributes is proposed:

1. **Predefinition:** the degree to which the process is predefined. Some processes are entirely predefined, meaning that every detail of the process has been defined prior to the commencement of its execution. During execution of the process, it is simply *enacted* according to its definition. Other processes are much less predefined, and are executed more in an ad-hoc fashion, where planning and execution converge (Moorman and Miner, 1998). For these kinds of processes, goals are usually more abstract and more subject to change.

2. **Determinism:** the degree to which the possible outcomes of the process can be determined in advance. Particularly in the area of workflow management, where processes are first modeled and encoded before they are enacted, the possible outcomes of a process are typically determined in advance. Highly deterministic processes produce results within a finite, well-defined set of possible outcomes. Other processes may produce much less predictable outcomes.

3. **Staticness:** the degree to which the way the process is performed is static. This applies mostly to predefined processes and is a measure of how much the execution of these processes changes over time from one enactment to another. Some processes are very static, being executed in the same fashion over a long period of time; others may be more volatile, changing constantly.

4. **Repetition:** the degree to which the process is repeated. Some processes are highly repetitive, being executed on a frequent basis. Others may be highly unique, being executed only once or only very sporadically.

Each of these attributes constitutes a continuum with values between its upper and lower extremes. Taken together, these attributes define a four-dimensional space of process types. Within it, a number of important process types stand out.

On the one hand, all attributes may be at their highest level, i.e. the process is entirely predefined, completely deterministic, completely static and highly repetitive. Such a kind of process is designated here as a *production process*. This type of process is very common in organizations and often constitutes high-volume core activities.

On the other hand, all attributes may be at their lowest level, i.e. the process is entirely non-predefined, non-deterministic, highly dynamic, and not repetitive. This kind of process is designated here as an *emergent process*. It has a goal that is typically expressed in more abstract terms, more like a mission statement rather than a detailed expression of a desired outcome, and which may mutate. An emergent process is controlled, or driven, by the increasing amount of knowledge available to it, which may cause its goal to be adjusted. This type of process occurs in many organizations, and is closely associated with innovation and improvisation (Moorman and Miner, 1998). Emergent processes are important as they facilitate organizational flexibility and are thus suited to organizations operating in volatile and competitive environments where flexibility is a key competency, as touched upon in the discussion in Chapter 1.

Another categorization of work processes has been proposed in (Hawryszkiewycz, 1999a), whose distinguishing attributes are the degree to which the tasks that make up a process are predefined, and the degree to which task sequences are predefined. This leads to the identification of following four types of processes, illustrated in Figure 2.1:

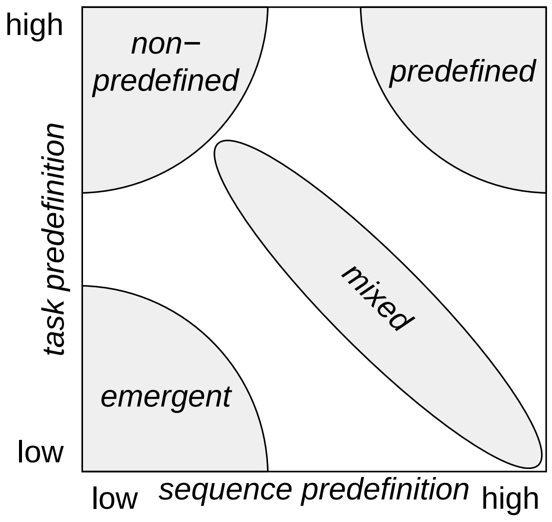


Figure 2.1: Process types related to task and sequence predefinition

1. **Predefined processes:** Both tasks and task sequences are predefined in detail. An example is a procurement process in a company where guidelines stipulate the specific tasks that need to be undertaken, as well as their sequencing.

2. **Non-predefined processes:** Tasks are pre-defined, but their sequence is decided during process execution. An example is the preparation of a report. Individual tasks such as writing report sections, preparation of diagrams, calculation of figures, etc. may be known in advance, but their sequencing may be flexible and is decided during process execution.

3. **Mixed processes:** These processes are a mix of predefined and non-predefined processes, with one type of process nested inside the other; e.g. a process where major steps are predefined, but where detailed tasks within each step are non-predefined. An example is software development. Major steps, such as analysis, design, coding, etc. are known in advance, but the actual tasks that need to be conducted during a given step, such as user interviews, prototype construction, etc. may only be decided sometime into a given process step.

4. **Emergent processes:** Neither the tasks nor their sequence are predefined and emerge during process execution. An example is the process of defining a new strategic direction for a company. Both the tasks that need to be undertaken, such as market analysis, focus group sessions, research into strategies of other companies, etc., as well as their sequencing may be unclear when the process is initiated and emerge only during its execution.

In the context of office work categorization, the degree of predefinition has been linked to task complexity (Picot and Reichwald, 1987, quoted in (Syring and Hasenkamp, 1997)). Three different types of office work are identified: (1) *unstructured tasks,* which are highly complex and allow for little prior planning; (2) *semi-structured tasks,* which have medium complexity and allow for some degree of prior planning; and (3) *structured tasks,* which have low complexity and offer a high possibility for prior planning. The authors go on to link information need, cooperation partners, and solution type to each of these types of office work: in unstructured tasks, information needs and cooperation partners are undetermined, as is the solution type; in structured tasks, all three are determined in advance; while semi-structured tasks lie somewhere in between (*ibid.*).

Thus, the degree of predefinition emerges as a common property in each of these definitions, allowing for a broad categorization of work processes, while the other mentioned properties allow for more fine-level classification.

2.1.1.2 Implications for Support Systems

Virtual collaboration processes are facilitated through computer-based systems. Different types of such systems are more appropriate for supporting different types of collaboration processes. Identification of the type of collaboration process to be supported therefore is important in identifying suitable support systems.

During the 1990's, workflow management systems have emerged as one such form of process support, and have subsequently enjoyed great success in the field. The Workflow Management Coalition, the relevant industry standards body, refer to a workflow management system as "a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic" (Hollingsworth, 1995). A workflow, in turn, is referred to as "the computerised facilitation or automation of a business process, in whole or part" (*ibid.*). Traditionally, workflow management systems have supported *production processes* as defined above (referred to as *predefined processes* in Hawryszkiewycz's classification, and *structured processes* according to Syring and Hasenkamp). Production processes need to be analyzed, modeled, and encoded in the workflow management system before they can be enacted through it. Their structure and possible outcomes are well-defined and subject to little change. Because of the effort involved in their analysis and modeling, it is usually only economical to do this for processes that are at least moderately repetitive.

Other types of processes, such as those that are not entirely deterministic, or those that are less repetitive, or more dynamic, are not well supported by conventional workflow technology. Consequently, the workflow community has, over the last few years, made efforts to make workflow systems more flexible and to support work processes which are not always well-defined or deterministic; see for instance (Casati and Pozzi, 1999; Sadiq, 1999). These have centred on either accounting for alternatives to standard workflows, handling workflow evolution, or dealing with exceptions as they occur. However, in most cases the basic model of a largely deterministic and knowable process has remained unchanged.

The category of emergent processes, on the other hand, is not well supported by workflow management systems. In emergent processes, details of process structure and of individual task goals may only be determined when the process is well underway, i.e. they *emerge* during process execution. Such processes are thus *improvisational* in nature (Moorman and Miner, 1998). They also tend to be highly knowledge-intensive, involving activities of knowledge sharing and creation, and may involve a considerable degree of tacit knowledge which can be difficult to encode (Nonaka, 1994). Such processes have been described in the literature as being "generally opportunistic in nature, result in disconnected and parallel work that must nevertheless be guided to a common goal" (Hawryszkiewycz, 1999b).

The features of emergent processes place unique demands on their support systems. In the context of process management, (Debenham, 1999) suggests that, given the special characteristics of emergent processes, any management system should *support*, rather than attempt to *control* emergent processes.

This research argues that the requirements of emergent processes are not met most appropriately by current workflow management systems, but instead by a class of groupware systems referred to here as *collaboration systems* and discussed in more depth in Section 2.2. In the context of collaborative research, a similar argument has been put forward (Appelt, 1999). Such collaboration systems need to be highly flexible to allow for evolution of the support provided along with the work carried out through them. As pointed out in (Hawryszkiewycz, 1999b), team formation and governance, provision of required knowledge sources, communication channels for geographically dispersed team members, and general tools all need to be supported. However, and more importantly, they must be supported in a way that facilitates flexible, easy evolution, thus aiding, and indeed enabling, process emergence.

Finally, the property of process emergence is not exclusively, or even primarily, dependent on certain features of the process itself, but includes the past experience of the team performing the process. That is, what is an emergent process to one team may be routine to another. When performed for the first time, the team may need to improvise to discover how best to perform the process. The next time such a process needs to be performed again, its performance can be based on the team's experience of the previous time and would be much less emergent. Once the team starts performing the process repeatedly, it becomes routine. That is, what started out as an emergent process has developed into a routine process, or conversely, a routine process may have started out as an emergent process. Supporting such processes that develop from being more emergent to being more routine thus puts additional demands on support systems.

2.1.2 Representing Virtual Collaboration Processes

Following the discussion of the kinds of virtual collaboration processes, this section investigates how such processes can be represented. Representing processes is important in order to facilitate communication about them, to serve as guidance for process execution, and to enable their analysis and comparison. Different kinds of information related to different aspects of processes can be represented. In the context of software development processes, for instance, (Curtis et al., 1992) identified four perspectives of processes which are commonly represented[1]:

1. **Functional perspective:** concerned with flows of information between tasks.
2. **Behavioural perspective:** concerned with aspects of the performance of processes such as sequencing, iteration, conditions, etc.
3. **Organizational perspective:** concerned with the assignment of members of an organization to tasks, the location of task performance, and the location for storing of objects involved in tasks.
4. **Informational perspective:** concerned with the structure of, and relationships (such as part-of, or version-of) between, informational entities (such as documents) involved in a process.

Representing any of these perspectives of a process involves the construction of a *process model*. A process model is defined as follows:

Definition 9 *A **process model** is an abstract description of an actual or proposed process (Curtis et al., 1992).*
□

[1]Note that the use of the term "task" used in the discussion that follows assumes the meaning as defined in this thesis, while (Curtis et al., 1992) refers to tasks as "process steps".

When a process model refers to an actual process, i.e. an instance of a process as it is being or has been executed, it is a *descriptive* process model; on the other hand, when it refers to a proposed process, it is a *prescriptive* process model.

A process model involves a number of basic *modeling elements*. A modeling element is an abstract representation of a type of real world entity involved in the process being modeled. Different process models may have different sets of modeling elements. For example, (Curtis et al., 1992) suggest the three modeling elements *agent* (a human or machine performing a task), *role* (a set of tasks assigned to an agent), and *artefact* (a product manipulated in a task). In the domain of virtual collaboration, however, usually other modeling elements are included as well, as discussed below.

Much work has been done on process modeling. Examples include: *Diplans*, which model coordination aspects of work processes (Holt, 1988; Holt, 1997); *FUNSOFT nets*, a type of Petri net developed to model software processes (Emmerich and Gruhn, 1991); the temporal logic-based *OBM* approach for step-wise refinement of organizational processes (Sa et al., 1993); and others. Recent years have seen an emphasis on the modeling of business processes, mostly in the context of business process re-engineering and workflow management. Examples of this kind of process modeling include: the *Actor Dependency Model*, which models networks of dependencies among organizational actors in the context of business process re-engineering (Yu and Mylopoulos, 1993); the *DEMO* approach which is based on Speech Act Theory and is used for business process redesign (Dietz, 1994); the *BPMAT* business process model (Bhaskar et al., 1994), used for analyzing and re-engineering processes; the *EPC* model* (Zukunft and Rump, 1996) for modeling of business processes for workflow support, which is based on the EPC model (Keller et al., 1992); the goal-based business process modeling approach of (Kueng et al., 1996); the event-based business process modeling approach of (Rohloff, 1996); and others.

However, the modeling of *virtual collaboration* processes has received very little attention, and consequently there are much fewer published modeling approaches. What sets collaboration processes apart from other types of work processes is that the interactions between humans are essential components of the process. In process models of face-to-face collaboration processes, where ample opportunity for such interactions exists, they are usually not represented. For models of virtual collaboration processes, however, these interactions need to be explicitly represented.

The remainder of this section introduces three modeling approaches which are concerned with modeling collaboration processes, one of which is specifically intended for modeling virtual collaboration processes. These include the following:

1. The Collaborative Process Model (Sarin et al., 1991).
2. The SeeMe modeling method (Herrmann et al., 2000).
3. Collaborative business process models (Hawryszkiewycz, 2000).

For each of these, a brief overview is given, including an identification of the modeling elements, and an example of the modeling notation.

2.1.2.1 Example Process

The various representations that follow are illustrated with the following example process, which deals with product concept development:

A manufacturing company finds its current product line is not performing as successfully in the market as it did in the past, and decides to develop new product concepts. A number of people with unique expertise are involved in this endeavour, each occupying a specific role in the overall process. They include product analysts who develop concepts for new products, market analysts who carry out market studies to evaluate the potential success of new product concepts, financial analysts who evaluate the financial feasibility of new product concepts, and a coordinator who oversees the entire process. The process involves these different roles in a number of tasks: an initial task of product brainstorming is followed by tasks of concept development, market study, and financial analysis, culminating in the end in a task of final report preparation where the outcome of the entire process is documented in a report that can later be submitted to upper management for approval. These tasks produce or manipulate a number of artefacts (documents): product ideas, product concepts, market analyses, product recommendations, financial analyses, and a final report. This process does not exist in isolation, but is followed by other processes, such as product design, manufacturing, etc.

2.1.2.2 The Collaborative Process Model

One of the earliest models explicitly concerned with modeling collaboration processes was developed at Xerox in the early 1990's (Sarin et al., 1991). In terms of Curtis *et al.*'s classification, this model provides primarily functional and behavioural perspectives of the processes modeled.

Collaborative Process Models represent *jobs* (which in terms of the terminology used here are the same as processes), where a job is defined as "a multi-person collaborative activity with some goal". A job includes users, documents, application tools, and other resources. Each job is composed of multiple tasks, and has associated with it a *workspace*

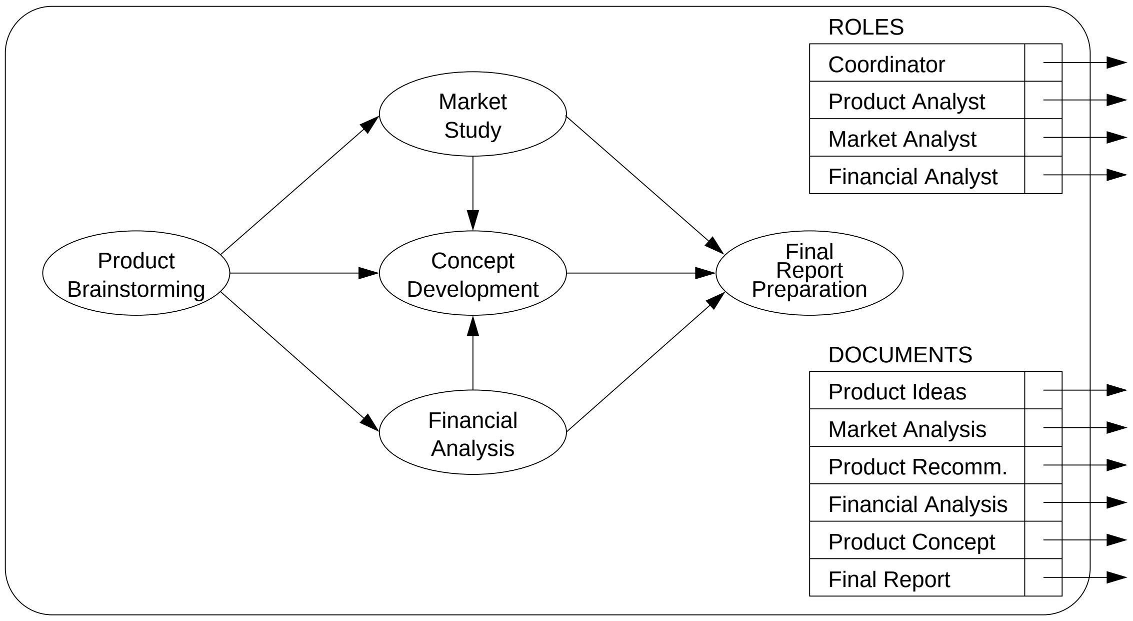


Figure 2.2: Collaborative Process Model of a product concept development process

within which activity related with the job is situated. Tasks are related through finish-to-start dependencies, which define a sequencing of task execution. Tasks may in turn be broken down into sub-tasks, resulting in a hierarchy of work activities.

When modeling collaborative processes, the following modeling elements are included, as defined in (Sarin et al., 1991):

1. **Documents:** “abstract data object[s] or resource[s] that [are] manipulated as a unit from tasks in jobs.”

2. **Roles:** “place holders for users who can perform tasks in the job.”

3. **Tasks:** “units of work.”

To illustrate this, a Collaborative Process Model of the job corresponding to the product concept development process described above is shown in Figure 2.2. Here, the whole process is shown as a network of dependent tasks, in this case consisting of five tasks, three of which are executed in parallel with some dependencies between them (the task Concept Development is partially dependent on output produced by tasks Market Study and Financial Analysis). The boxes at the top right and bottom right show, respectively, the externally visible roles and documents involved in the process.

To show details of individual tasks, any task in the Collaborative Process Model may be decomposed into sub-tasks, which are shown in a separate diagram. For instance,

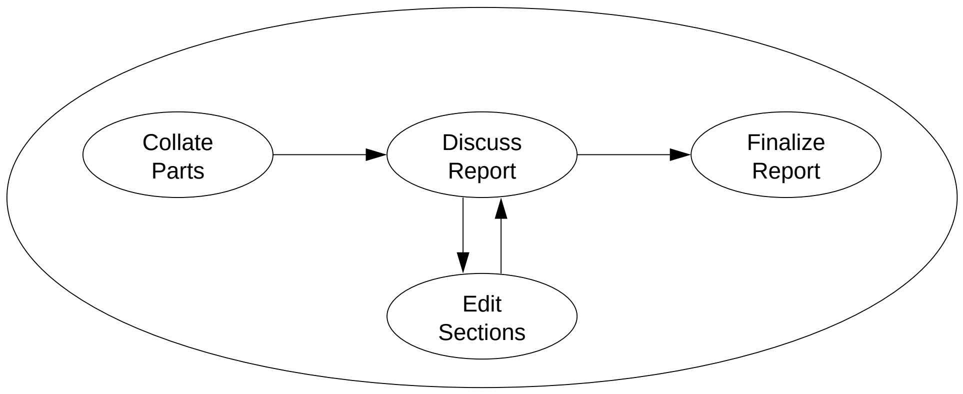


Figure 2.3: Decomposition of the Final Report Preparation task from the Collaborative Process Model of a product concept development process

details of the Final Report Preparation task are shown in Figure 2.3. This task, as modeled here, consists of four sub-tasks: first the different sections of the report are collated in the proper sequence, then this draft version of the report is discussed, if necessary any sections of the report are edited, and lastly the report is finalized. Note that the editing of sections and discussing of the draft report may be repeatedly performed, in a loop.

While the Collaborative Process Model is intended to represent collaborative processes, interactions among the actors involved in the collaboration are not represented. Only the modeled dependencies among tasks imply the likelihood of interactions among the task performers. Thus the model is very limited in its ability to represent highly collaborative processes, and resembles more workflow-oriented process models.

2.1.2.3 SeeMe Models

SeeMe models describe semi-structured socio-technical systems. The SeeMe diagramming technique was developed to aid requirements engineering and system design, and to allow an element of vagueness, incompleteness and contradictions to be explicitly represented (Herrmann et al., 2000). In terms of Curtis *et al.*'s classification, SeeMe models can provide functional, behavioural, organizational, and informational perspectives. Most commonly, however, the emphasis is on the functional and behavioural aspects of processes.

SeeMe models are made up of following three basic modeling elements:

1. **Roles:** "a set of rights and responsibilities assigned to a person, a group or an organizational unit" (Herrmann et al., 2000).

2. **Entities:** passive objects or things used as resources in activities.

3. **Activities:** actions performed by people occupying roles, which act on entities.

Any type of modeling element can be related to any other type of modeling element through default semantics. For instance, a role can be related to an activity, meaning that the role carries out the activity. Another example is where an activity is related to another activity meaning that one is followed by the other. However, these semantics are flexible and can be extended as needed.

Any of the modeling elements can be at any desired level of granularity. For instance, an entire process, a task, a sub-task, or a minute user action can all be represented as activities. The same applies to entities and roles. Furthermore, hierarchical structuring of any type of modeling element can be represented in a given model through *nesting* of the detail level inside the higher-level element. Finally, relationships between diagrams can be represented in *metadiagrams*. Thereby a complex process can be modeled as a collection of separate diagrams, with one metadiagram establishing how these diagrams are related to one another.

SeeMe models can also include conditions and branches to represent details of behaviour. In order to accommodate vagueness or ambiguity, certain notations exist to indicate that information may be incomplete, may have been purposely omitted (such as when it is deemed not to be relevant), or that its correctness is doubtful.

An illustration of the use of SeeMe models is given in Figure 2.4, which shows a model of the product concept development process introduced earlier. Depicted are roles in ovals, activities (in this case tasks) in boxes with rounded corners, and entities (documents) in rectangles. Arrows express relationships: arrows pointing from roles to activities mean that the role carries out the activity; arrows between activities signify sequencing; and arrows between activities and entities mean that the entity is used in, and/or created/changed by the activity.

Nesting is applied in order to aggregate roles, activities, and entities, thereby simplifying the diagram. When modeling elements are nested, any arrow connecting the compound modeling element to another modeling element represents the same type of arrow connecting each of its contained modeling elements with the other modeling element. Thus, for instance, the roles Market Analyst, Product Analyst, and Financial Analyst are aggregated into the role Writer. This aggregated role carries out the Product Brainstorming and Final Report Preparation activities, meaning that each of the three roles contained in the role Writer is involved in these two activities. Similarly, the three activities Market Study, Financial Analysis, and Concept Development are aggregated to a compound activity which, as a whole, is successor to activity Product Brainstorming and is followed by activity Final Report Preparation. This compound activity receives the

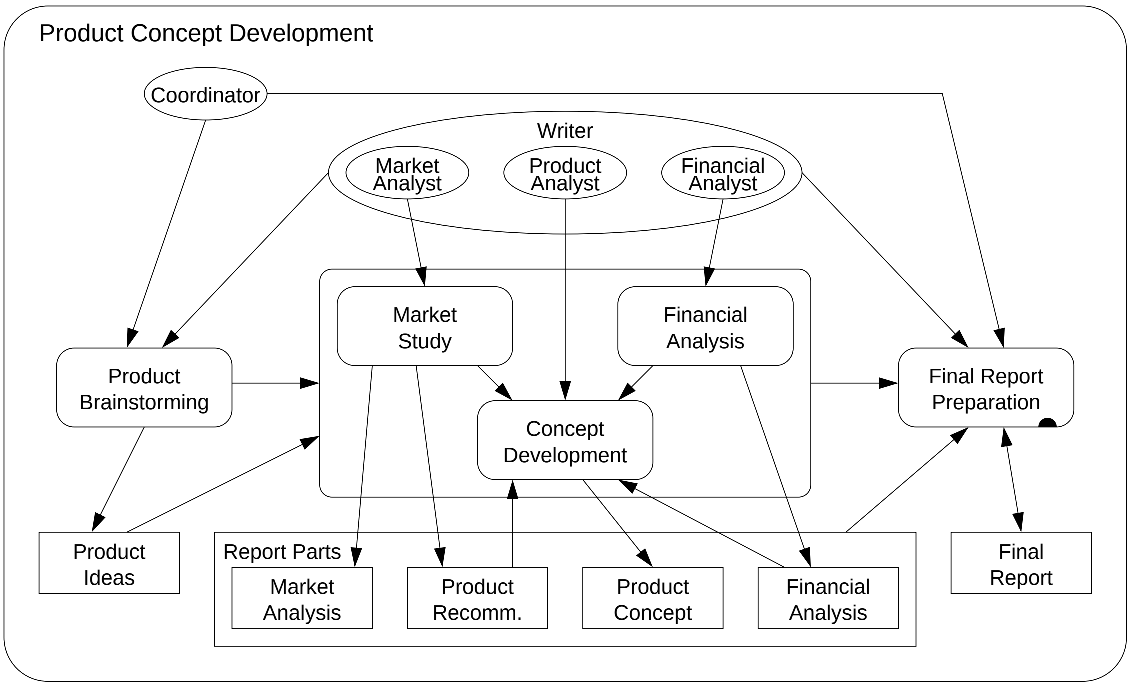


Figure 2.4: SeeMe model of a product concept development process

entity Product Ideas, meaning that this entity is used by each of the compound activity's constituent activities. Finally, the four entities Market Analysis, Product Recommendations, Product Concept, and Financial Analysis are nested inside the entity Report Parts. This compound entity is used in the activity Final Report Preparation, meaning that each of the contained entities is used by this activity. Thus the nesting of modeling elements enables a convenient shorthand in modeling relationships between these and other modeling elements.

In the Final Report Preparation activity, an intended omission of detail is indicated by the black filled semi-circle near its lower right corner: it denotes that more detail about this activity is available but has not been included in the diagram, and has instead been placed in another diagram. Figure 2.5 shows the detail of the Final Report Preparation activity as a separate SeeMe model. It can be seen that the modeling of this task is very similar to that of the entire process, i.e. again roles, activities and entities are represented, and are linked through arrows signifying relationships among them. The diagram also shows a condition, in the hexagon linking the Discuss Report and Edit Sections activities: if in the Discuss Report activity a section of the report is found to require revision, the Edit Sections activity is performed. This may be repeated any number of times, each time updating the draft report, until no more revisions are required, at which time the Finalize Report task is performed, producing the Final Report entity. As the entity Draft Report

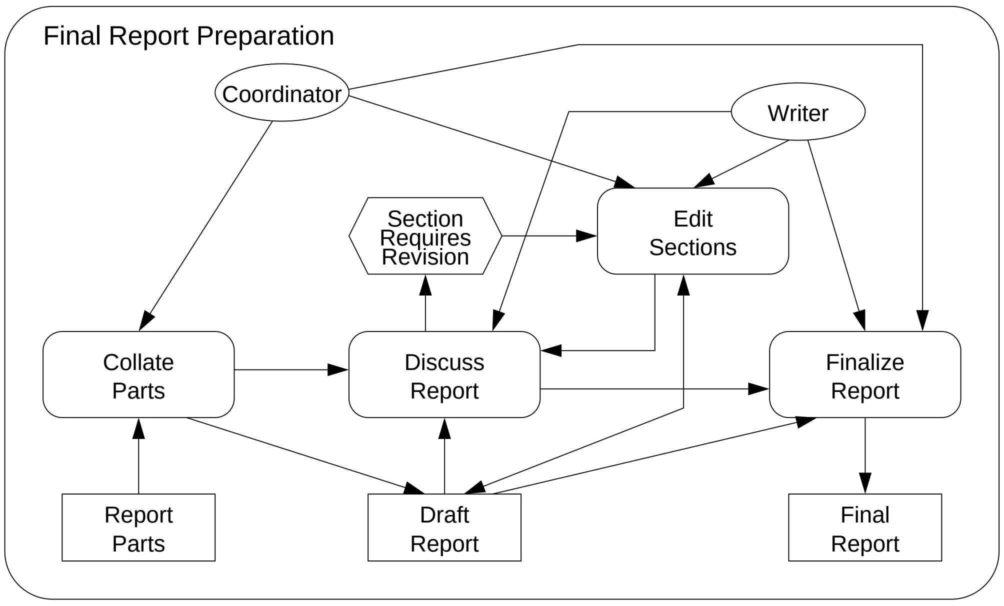


Figure 2.5: SeeMe model of the Final Report Preparation task of a product concept development process

is internal to the Final Report Preparation activity, it does not appear in the model of the whole process in Figure 2.4.

SeeMe models are intended to provide a semi-formal, flexible notation that allows vagueness inherent in socio-technical systems to be made explicit, rather than enforcing completeness of specification. As stated by their authors, SeeMe diagrams are seen as a modeling notation for the representation of cooperation and communication processes (Herrmann et al., 2000). Given the inclusion of details on roles involved in activities, SeeMe diagrams appear to be more appropriate for representing collaboration processes than Collaborative Process Model diagrams.

2.1.2.4 Collaborative Business Process Models

A model for representing collaborative business processes was introduced in (Hawryszkiewycz, 2000). In terms of Curtis *et al.*'s classification, this model provides primarily functional and behavioural perspectives of the processes modeled.

The collaborative business process model is part of a methodology for the design of collaborative applications. These applications are intended to be used by virtual teams performing the modeled processes in a mode of virtual collaboration. Support for emergent processes, and for the evolution of processes over time, are seen as key requirements. Thus, modeling represents processes in terms of main process elements and their relationships, rather than representing fine detail which is likely to change as processes evolve.

Collaborative business process models contain following four modeling elements:

1. **Roles:** organizational roles with associated rights and responsibilities.

2. **Artefacts:** passive objects, such as documents, used or manipulated in activities.

3. **Activities:** tasks performed by people occupying roles, and affecting artefacts.

4. **Discussions:** communication channels between people, involving one or more roles.

The first three modeling elements are similar to those of other modeling approaches. The last one, discussions, however, is unique to collaborative business process models. Discussions represent communicative interactions between people. In conventional processes these need not be explicitly represented, as face-to-face settings provide ample opportunity for communication between people, both formal and informal. However, since this modeling approach specifically focuses on virtual collaboration, channels allowing for the communication between people need to be explicitly represented.

Collaborative business process models include diagramming notations both for representing the overall structure of virtual collaboration processes, and for representing details of individual tasks. Processes are represented in a notation called *Rich Pictures*. The rich picture is a modeling tool originally introduced by the Soft Systems Methodology (SSM) (Checkland, 1981). It captures relationships and connections between elements of human activity. However, the modeling method of Hawryszkiewycz has proposed a modified form of rich pictures, which constitutes both an augmentation, as well as a simplification of the original form (Hawryszkiewycz, 2000). Unlike the original rich pictures, it introduces standard symbols and has a defined syntax and semantics. An example of a rich picture of the product concept development process presented earlier is shown in Figure 2.6. Here roles are shown as stick figures, activities as clouds, and artefacts as boxes. Lines and arrows relate the different modeling elements to each other; for instance, lines linking roles and activities represent involvement of roles in the activities, while arrows between artefacts and activities represent the use or creation of the artefacts in the activities.

Hawryszkiewycz-style rich pictures provide a functional perspective of the overall process. As an augmentation to traditional rich pictures, the collaborative business process modeling approach is complemented by so-called *transition diagrams*, which express sequencing of activities: transition diagrams capture the possible transitions between activities in a rich picture. For example, a transition diagram corresponding to the

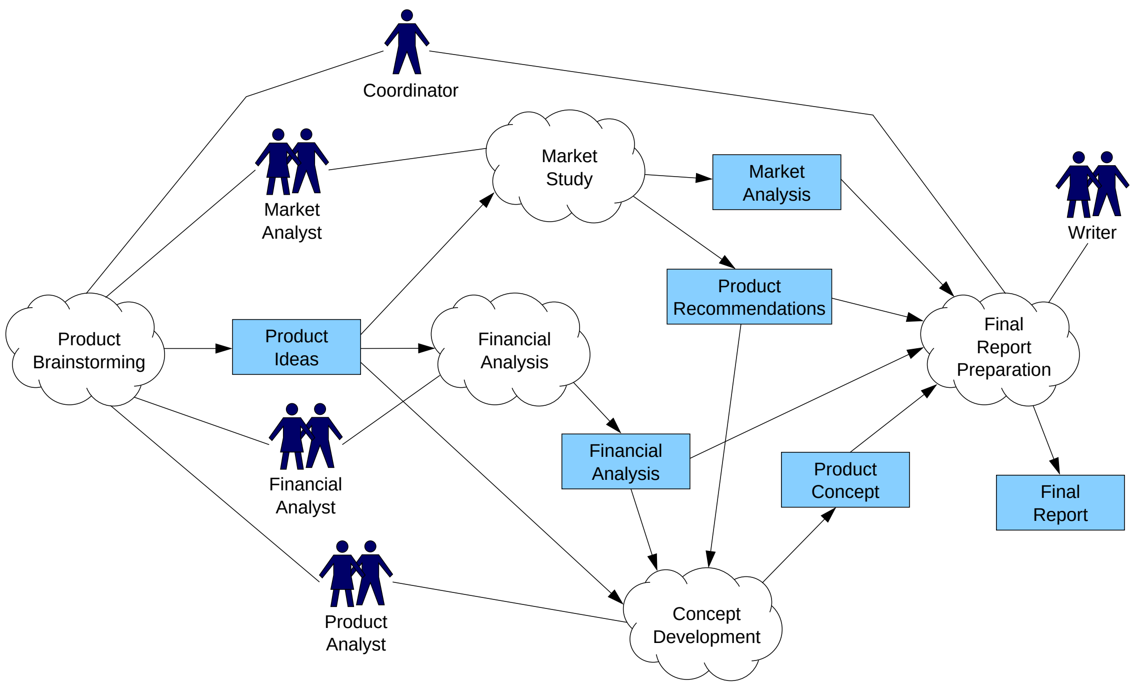


Figure 2.6: A Hawryszkiewycz-style rich picture of a product concept development process

rich picture of Figure 2.6 is shown in Figure 2.7. These transition diagrams resemble the task network of Collaborative Process Models, such as the one shown on page 23. Each of the activities in a rich picture and its corresponding transition diagram represents one task. When tasks are very large, however, they may need to be decomposed by modeling them as a collection of related sub-tasks. To allow the representation of such sub-tasks, Hawryszkiewycz-style rich pictures can be expanded to multiple levels of detail, much like data flow diagrams, by modeling details of one higher-level activity as an entire lower-level rich picture consisting of sub-activities. Each such expansion is accompanied by its own separate transition diagram.

The internal structure of tasks that are sufficiently small, i.e. which have been decomposed to a level where further decomposition is not deemed necessary, is represented using another modeling notation, the so-called *MOO diagram* (Hawryszkiewycz, 2000). A MOO diagram represents internals of activities as a combination of roles, artefacts, and discussions. At this level, communicative interactions are made explicit. An example MOO diagram corresponding to the Final Report Preparation task of Figure 2.6 is shown in Figure 2.8. Here, the hexagon represents a discussion, ovals roles, and boxes with rounded corners artefacts. Arrows represent involvement and use/creation, respectively, as above with rich pictures.

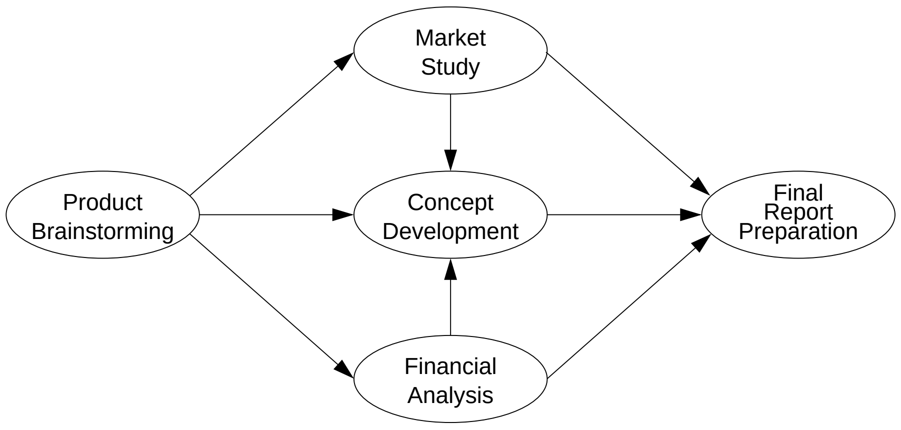


Figure 2.7: Transition diagram for a product concept development process

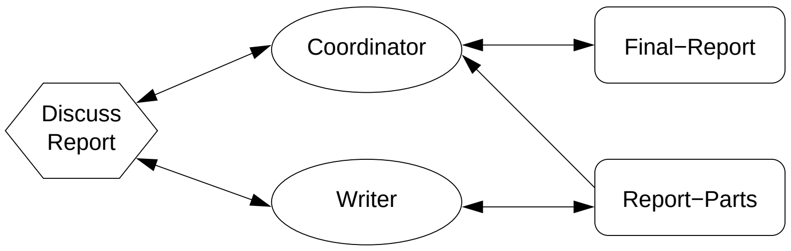


Figure 2.8: MOO diagram for the Final Report Preparation task of a product concept development process

Compared with other modeling notations, collaborative business process models are more abstract, as they are not intended to model a great level of detail of processes. By outlining only general process elements and leaving open the actual implementation of the process, this notation is particularly suited for representing partially planned and emergent processes. Moreover, the explicit representation of communicative interactions makes this modeling notation well suited for representing virtual collaboration processes.

———— · ————

The three presented process modeling approaches are an illustration of some of the ways that collaborative processes can be represented. While all of them allow both processes and their constituent tasks to be modeled, the modeling differs in terms of the modeling elements that are included, the modeling notation, the amount of detail represented, and whether or not communicative interactions are represented. In later chapters, some of these notations are used for the representation of virtual collaboration.

2.2 Collaboration Systems

The preceding section discussed virtual collaboration processes. The present section carries on from there to investigate the systems through which these processes are supported and carried out.

Physical collaboration takes place in physical space where all necessary tools, objects and people are assembled for the task at hand. A product concept development process, for example, may bring together experts comprising product analysts, market analysts, financial analysts, as well as others; documents in various stages of development; product catalogues, design manuals, company guidelines, various reference works, word processing tools, calculators, etc. into a single office or a set of offices that are in reasonably close proximity to one another to afford the opportunity of easy face-to-face interaction.

If face-to-face contact is not an available option, however, the carrying out of collaborative tasks, such as the ones involved in product concept development, becomes challenging. Collaboration then becomes possible if it is mediated by technology, when it is referred to as *remote collaboration*, or more commonly *virtual collaboration* (cf. the definition of virtual collaboration on page 3). Moreover, different types of technology provide different degrees of support and are suited to different kinds of collaborative tasks.

For example, in the simplest case, the telephone can be used to support the collaborative team's communication requirements, specifically synchronous communication, while fax allows for primitive document exchange. Video-conferencing allows two or more participants to communicate simultaneously, but can usually only be used effectively among a limited number of participants. For asynchronous communication, electronic mail can be used, which also offers a somewhat more sophisticated document exchange potential, when compared to the alternatives of fax or postal mail, through the use of email attachments.

However, all of these technologies provide only partial support for the overall requirements of collaborative groups, which have been termed the three 'C's of group interaction: communication, collaboration, and coordination (Ellis et al., 1991).

Systems which attempt to offer a more complete support for the different aspects of virtual collaboration provide integrated *environments* in which collaboration can take place, not merely a collection of separate tools. These kinds of environments have been named differently in the literature: *collaboration spaces* (Farshchian and Divitini, 1997), *collaborative environments* (Farshchian and Divitini, 1997; Simoff and Maher, 2000), *collaborative virtual environments* (Benford et al., 1997), *collaborative virtual workspaces* (Spellman et al., 1997), *collaborative workspaces* (Pankoke-Babatz and Syri, 1997), *locales* (Fitzpatrick et al., 1996), *network places* (Roseman and Greenberg, 1996), *shared workspaces* (Dourish and Bellotti, 1992; ter Hofte et al., 1996), *teamrooms* (Roseman and Greenberg, 1996), *virtual collocation environments* (Poltrock and Engelbeck, 1997), *virtual information spaces* (Rhee, 1999), and *workspaces* (Baker et al., 1999). Common to research and development effort on such environments is the desire to provide the opportunity for bringing together required people, resources, and communication channels for joint activity in a virtual place or set of places. For the remainder of this thesis, the term *collaboration space* shall encompass these various terms, while a system that provides collaboration spaces shall be referred to as a *collaboration system*. These two terms are defined as follows:

> **Definition 10** *A **collaboration space** is a virtual space which provides the opportunity for bringing together people, artefacts, and communication channels for individual or joint activity.*
> □

> **Definition 11** *A **collaboration system** is a software system which supports virtual collaboration through the provision of collaboration spaces.*
> □

2.2.1 Structuring Metaphors

As the generality of collaboration systems aim to provide some kind of *environment* for collaboration, the question arises how this environment should be internally structured, and what metaphors should be employed for representing and presenting the virtual environment.

Existing collaboration systems can be broadly divided into two general categories with regard to the structuring metaphor employed: (1) systems which employ a spatial metaphor; and (2) systems which employ an abstract metaphor.

The first category of systems conceive of some kind of virtual counterpart to a physical structure. For instance, some systems provide a set of rooms that are connected by doorways (Roseman and Greenberg, 1996); different rooms are used for different tasks or by different users, and users move from room to room to work on different tasks or to interact with different users. Other systems extend this metaphor to entire buildings which are organized into different floors, have corridors, and rooms for different purposes. The way this spatial metaphor is represented to the user ranges from the very simple, textual interface in the style of MUDs (multi-user domains) (Churchill and Bly, 1999); to the two-dimensional layout of objects in a “room”, with simple clickable lists of rooms or doorways to other rooms; to sophisticated three-dimensional virtual reality worlds (Greenhalgh, 1999) in which the user may be represented by an avatar or other representation (Çapin et al., 1999), and where navigation from room to room is visualized as the actual action of walking out one door, down a corridor, and through another door into a different room.

The second category of systems, the ones employing an abstract metaphor, do not borrow concepts from real-world structures. Instead, the virtual environment is presented as a kind of abstract entity, a cyberspace with no connection to the physical world. These environments may be referred to within the collaboration system as *workspaces*, *virtual places*, or other terms mentioned earlier. As with the systems based on spatial metaphors, these systems too usually allow users to set up different environments for different tasks or for use by different users. Navigation between different environments, however, here follows different metaphors, such as that of a tree, or web. Representation of the environment to the user is typically as a screen within which objects such as documents and tools are listed or displayed in textual and/or graphical form. Users are usually not represented through embodiments in these systems, and the user perspective is thus more detached from the virtual space when compared with the spatially-oriented systems.

There has been some discussion in the literature on the appropriateness of applying spatial metaphors to virtual environments such as collaboration spaces, see for instance (Harrison and Dourish, 1996). These authors emphasize the importance of *place* rather than *space* in collaboration systems: *"Space is the opportunity; place is the understood reality"* (Harrison and Dourish, 1996, p. 69). That is, what a place *is* is not so much a matter of its spatial features but rather what its users *do* within it and which turns a space into a place (this is similar to the difference between a house and a home). Thus it is important that any collaboration space creates the opportunity for users to appropriate it and turn it into a place for collaboration.

2.2.2 Awareness

Face-to-face collaboration affords the opportunity for a great amount of peripheral perception, i.e. the ability to know what is "going on" around oneself without actively seeking this knowledge. This includes knowledge about who is around, what they are doing, who is currently working with whom, as well as overhearing and overseeing the conversations and work of others. The importance of this *awareness* of the activities of others in facilitating the individual's work as part of an overall collaborative task becomes particularly evident when it is absent, such as in virtual collaboration settings. It has been argued that the opportunity for casual interaction is of great importance in facilitating collaboration (Kraut et al., 1988). Awareness of others is a prerequisite for making such *opportunistic interactions* possible.

In the context of virtual collaboration, awareness has been defined as “an understanding of the activities of others, which provides a context for your own activity” (Dourish and Bellotti, 1992). As the kind of awareness afforded by face-to-face collaboration is absent in the realm of collaboration spaces, a substitute has to be explicitly provided. The earliest work on providing awareness in CSCW systems dates back to the early 1990’s and the research of a group at Rank Xerox EuroPARC (Moran and Anderson, 1990; Borning and Travers, 1991; Dourish and Bellotti, 1992; Dourish and Bly, 1992). There, awareness was provided by maintaining audio and video links among a number of locations. The intention was to provide a constant stream of background information that could be tapped into as and when needed. To quote from Moran and Anderson:

> . . . people deal with a complex environment by not attending to most of it most of the time. It is important not to saturate people with things they cannot ignore. Our approach to using audio technology to provide a natural ambient audio environment . . . illustrates the principle. On the other hand, people are very aware of what goes on in their environment; without such awareness they would feel isolated. The environment needs to be rich with many things (including other people) that could be attended to. The environment needs to signal the availability of these things by tapping on people’s ability to peripherally process the non-attended parts of the environment so that they can redirect their attention when appropriate. (Moran and Anderson, 1990, p. 386)

It has been suggested that four different types of awareness can be distinguished (Gutwin and Greenberg, 1995):

1. *Informal awareness*: knowledge of who is around and what they are doing.
2. *Social awareness*: knowledge of the social and conversational context.
3. *Group-structural awareness*: knowledge of people’s roles and responsibilities, as well as group processes.
4. *Workspace awareness*: knowledge of others’ interactions with the virtual space and its artefacts.

However, this list of types of awareness is far from exhaustive, and other types could be distinguished; for instance *cultural awareness*, i.e. the knowledge of cultural norms of the people involved in the collaboration, and thus expectations of how to behave.

Out of the above list, the first three are general awareness categories that apply to both face-to-face and virtual collaboration. The fourth category, however, applies exclusively to virtual collaboration, and consequently CSCW research has put the greatest emphasis on it.

Workspace awareness aims to assist users in acquiring knowledge on five aspects of their collaboration space: *who* (is around), *where* (are they working, looking), *what* (are they doing and working on), *when* (did something happen), and *how* (did something happen) (Gutwin and Greenberg, 2002). This kind of awareness knowledge is seen as having following uses in collaboration (*ibid.*):

- simplification of communication
- coordination of actions and activities
- anticipation of events
- provision of assistance
- management of coupling

This research activity has contributed to a richer support of awareness in groupware systems. Among the most active in this area, the University of Calgary's Grouplab has been investigating awareness and building prototypes, particularly in the realm of synchronous collaboration, of a number of *awareness widgets* (Gutwin et al., 1995). These are interface components that contribute to a certain aspect of awareness. Navigation of the virtual environment, for instance, is supported by awareness widgets that provide *secondary viewports* or *miniature views*; artefact manipulation is made visible through *action indicators*; characteristics and progress of an action are shown through *action animations* and *sound cues*; and *alternative view representations* provide gestural communication and deictic references (Gutwin and Greenberg, 1998a; Gutwin and Greenberg, 1998b).

This work on awareness in synchronous collaboration has been extended by others to include asynchronous modes of collaboration (Dourish, 1997). Dourish argues that collaborative interaction progresses through repeated cycles of divergence and synchronisation, aided by awareness information. The difference between synchronous and asynchronous collaboration lies in the timescales of these cycles: seconds in the synchronous case, versus hours or days in the asynchronous case. As a consequence, the focus of awareness shifts: in synchronous collaboration the focus is mainly on other group members and their actions, while in asynchronous collaboration the focus is on the artefact and its change over time. Awareness thus extends to cover the history of collaboration, rather than only the here-and-now.

Recent work that has been aimed at providing richer resources for awareness and collaboration has combined both synchronous and asynchronous awareness with communication facilities and media spaces (Greenberg and Rounding, 2001). On the other hand, the scope of awareness has been extended to refer to a knowledge of more general features of the collaboration at hand, such as the collaboration's goal, priorities, and milestones, as well as a shared terminology relating to the collaboration (Hawryszkiewycz, 1999b).

This discussion on awareness has highlighted the importance of the concept in the support of virtual collaboration. While most existing collaboration systems include support for awareness, they differ sometimes significantly in the extent to which they provide this support, as well as the type of awareness they offer. In the review of a number of collaboration systems that follows below, a discussion of their support of awareness is included to explore this aspect further.

2.2.3 Review of Existing Collaboration Systems

Today, there exist a great many systems that can be referred to as collaboration systems, according to the definition of the term given above (cf. p. 32). Originally, such systems were the products of research laboratories and universities, however in recent years there has been a proliferation of commercial collaboration systems, particularly those that are web-based. The intention of this section is not to exhaustively review all of these systems, a near-impossible task considering the pace with which these systems appear on the market, as well as vanish. Rather, the intention is to give a flavour of some of the systems that have been developed, to highlight the variety of features with which these systems attempt to support virtual collaboration. The emphasis here is on systems that have come out of research and which are documented in the research literature. Some of these, however, have since evolved into commercial products.

The following collaboration systems are reviewed, listed in chronological order of their first mention (shown below in parentheses):

- BSCW (1995)
- CBE (1996)
- TeamRooms (1996)
- CVW (1997)

- ORBIT (1997)
- LIVENET (1999)

In the following subsections, these systems are briefly introduced and their main features discussed. One of these systems, LIVENET, is treated in slightly more detail, as later chapters use this system for illustration.

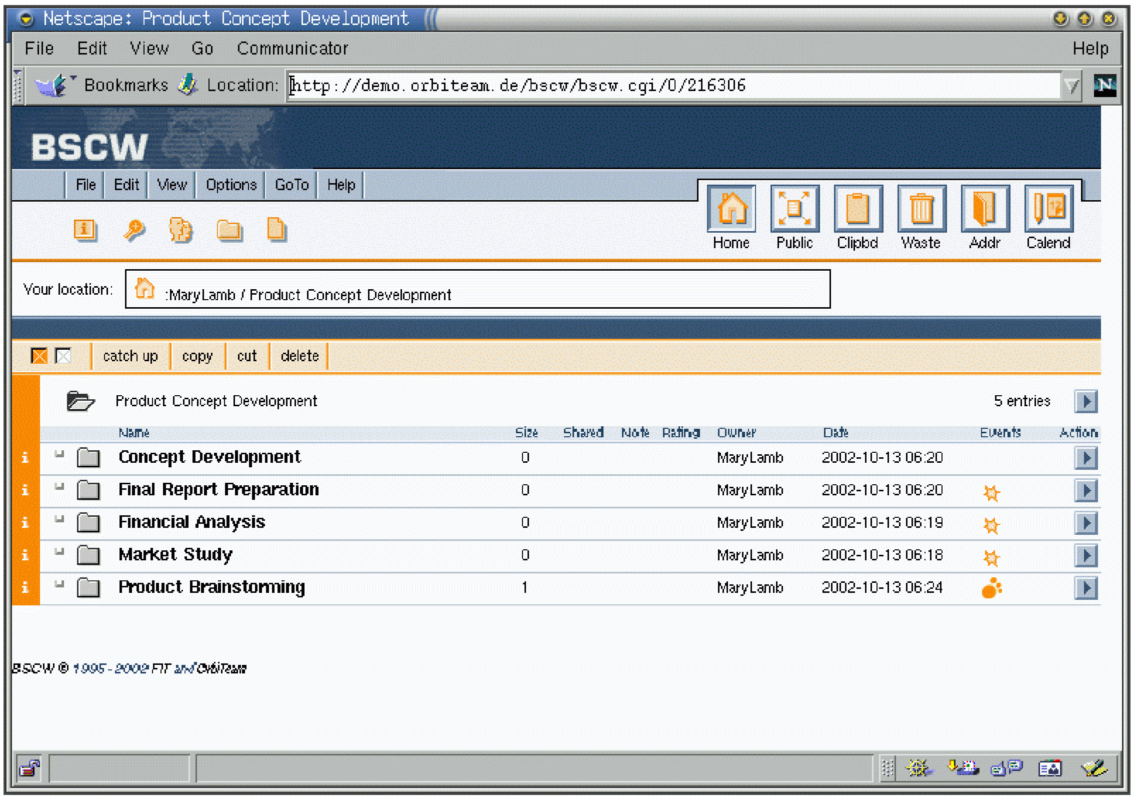
2.2.3.1 BSCW

The BSCW system was developed in the mid-1990's at the GMD in Germany, and has since undergone continuous further development (Bentley et al., 1995; Bentley et al., 1997; Appelt, 1999). It was originally conceived as "a means of supporting the work of widely-dispersed work-groups, particularly those involved in large research and development projects" (Bentley et al., 1995). The original motivation for developing the BSCW system was the limitation of existing technologies such as email and ftp that only allow information *exchange*, while there was the perceived need for collaborative information *sharing*. Information sharing is understood to involve not only the ability to access a shared copy of a document, but also to make annotations, and to see details of changes made by other users.

The BSCW system is web-based: a central web server is accessed by multiple clients using standard web browsers. The user interface is primarily text-based, complemented by a few graphical icons. The interface is similar to that of a file system browser, made up of a hierarchy of folders containing objects (such as other folders or documents). A sample screen of a BSCW workspace is shown in Figure 2.9. Documents can be locked, multiple versions can be maintained, and annotations can be attached to documents. While the emphasis of the system lies in (primarily asynchronous) information sharing, the use of other collaboration tools is made possible through interfaces to external synchronous conferencing or shared whiteboard applications. A built-in discussion facility for asynchronous communication also exists, and newer versions of BSCW have added support for simple project management and calendaring, as well as for stored searches of information within BSCW and on the Internet.

Structuring Metaphor

BSCW uses an abstract structuring metaphor resembling that of a file system. Individual collaboration spaces are referred to as *workspaces*. Each workspace has a *root folder*, which may contain various objects such as documents, links (URLs), as well as other folders. Those folders in turn may contain other folders, resulting in a tree-structured


Netscape: Product Concept Development
File Edit View Go Communicator Help
Bookmarks Location: http://demo.orbiteam.de/bscw/bscw.cgi/0/216306
BSCW
File Edit View Options GoTo Help
Home Public Clipbd Waste Addr Calend
Your location: :MaryLamb / Product Concept Development
catch up copy cut delete
Product Concept Development
5 entries
BSCW © 1995-2002 FIT and OrbiTeam

Name	Size	Shared	Note	Rating	Owner	Date	Events	Action
Concept Development	0				MaryLamb	2002-10-13 06:20		
Final Report Preparation	0				MaryLamb	2002-10-13 06:20	☼	
Financial Analysis	0				MaryLamb	2002-10-13 06:19	☼	
Market Study	0				MaryLamb	2002-10-13 06:18	☼	
Product Brainstorming	1				MaryLamb	2002-10-13 06:24	●	

Figure 2.9: BSCW user interface

folder hierarchy. Other types of objects that may be contained in a folder are projects, calendars, discussion forums, and stored searches.

Awareness Support

As BSCW is a system for primarily asynchronous collaboration, awareness is provided in the form of a history of events. For instance, whenever an object is added to a folder, or read or changed, this is shown in the user interface through icons indicating that such events have transpired. The user can then click on the icon to obtain more information on the event. In this way, a basic form of awareness about activities on the objects within the current view of the user is provided. Additionally, users can choose to receive an emailed daily report of events to become aware of what events have taken place in the workspace as a whole during a given day.

Access Control

BSCW provides fine-level access control, where detailed access privileges can be assigned for every object and user in a workspace. This is complemented by role-based access control, allowing roles to be created, and access permission templates to be defined for each role. When users are added to a specific role, they inherit all the permissions which that role possesses.

2.2.3.2 CBE

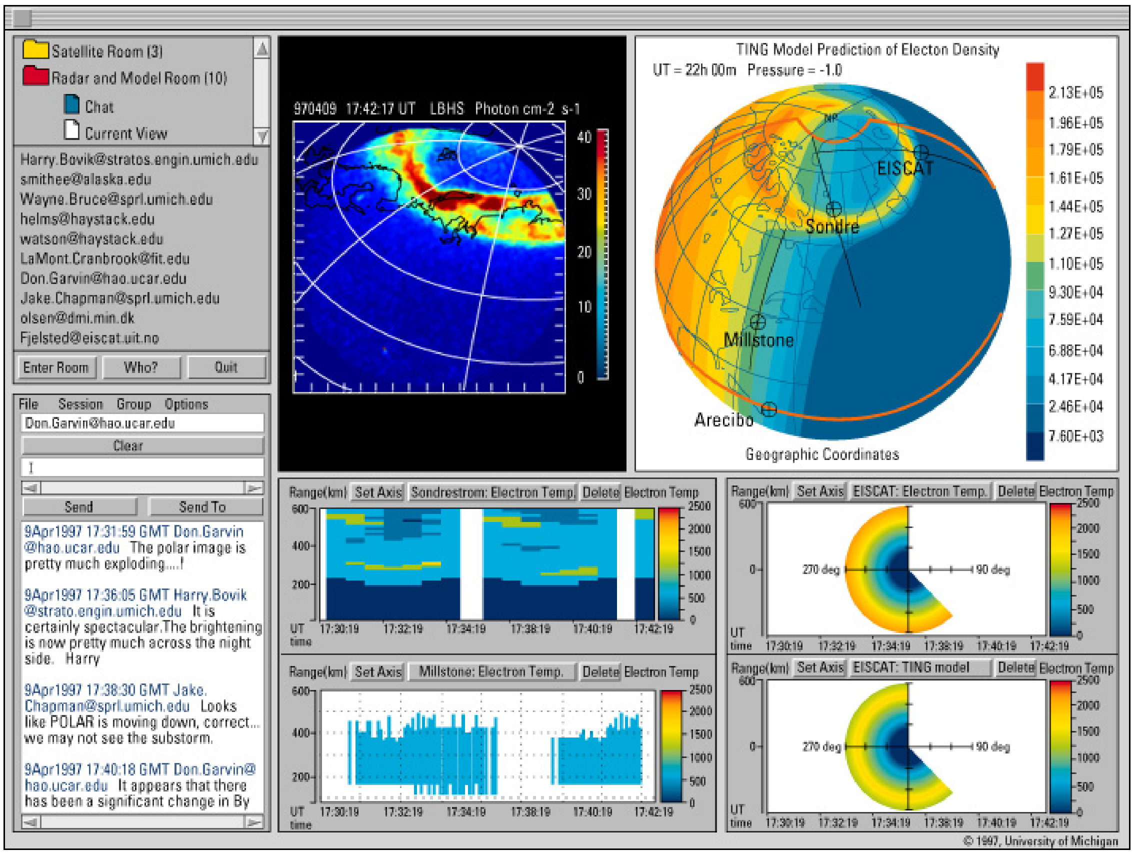
CBE is a toolkit for creating extensible collaboration environments in Java, created at the University of Michigan in 1996 (Lee et al., 1996). It is an outgrowth of the UARC (Upper Atmospheric Research Collaboratory) scientific collaboration system (Clauer et al., 1995). Using CBE, collections of loosely-coupled tools can be assembled to support specific tasks. Its four main design components are *applets*, *users*, *applet groups* and *rooms*. Applets are small client-side applications that may run in a web browser but could also be local applications on the user's computer. They may either offer general collaboration services, such as shared whiteboards or multi-user chat; or be domain-specific, such as scientific visualization tools. Applet groups consist of all instances of the same type of applet belonging to users in the same room, among which data is shared. Both individual applets, as well as entire applet groups can be dynamically transferred between rooms. Individual and group work are supported through private and group rooms. As room state persists across sessions, CBE can be used for both synchronous and asynchronous work. By allowing users to create their own applets, CBE is user-extensible. Figure 2.10 shows an example of a number of generic and domain-specific applets being used by a number of collaborating users.

Structuring Metaphor

In CBE, shared workspaces called *rooms* provide the basic structuring construct. However, the term is used in a somewhat abstract sense, and is not intended to imply a strict room-based metaphor. Rooms provide a space for users to meet, and rooms house the applets which users use. A room thus constitutes the working environment for an individual or a group of collaborating users. Rooms cannot be nested, but may be linked to each other, resulting in a non-hierarchical network of collaboration spaces.

Awareness Support

Basic awareness is provided in the form of information about users who are present in a room. In the case of synchronous collaboration, information about the actions of other users may be provided by the applets used, but this is dependent on each such applet.



Harry.Bovik@stratos.engin.umich.edu
smithee@alaska.edu
Wayne.Bruce@sprl.umich.edu
helms@haystack.edu
watson@haystack.edu
LaMont.Cranbrook@fit.edu
Don.Garvin@hao.ucar.edu
Jake.Chapman@sprl.umich.edu
olsen@dmi.min.dk
Fjelsted@eiscat.uit.no

9Apr1997 17:31:59 GMT Don.Garvin@hao.ucar.edu The polar image is pretty much exploding....!

9Apr1997 17:36:05 GMT Harry.Bovik@strato.engin.umich.edu It is certainly spectacular.The brightening is now pretty much across the night side. Harry

9Apr1997 17:38:30 GMT Jake.Chapman@sprl.umich.edu Looks like POLAR is moving down, correct... we may not see the substorm.

9Apr1997 17:40:18 GMT Don.Garvin@hao.ucar.edu It appears that there has been a significant change in By

Satellite Room (3)
Radar and Model Room (10)
Chat
Current View
Enter Room
Who?
Quit
File Session Group Options
Don.Garvin@hao.ucar.edu
Clear
Send
Send To
970409 17:42:17 UT LBHS Photon cm-2 s-1
TING Model Prediction of Electon Density
UT = 22h 00m Pressure = -1.0
NP
EISCAT
Sondre
Millstone
Arecibo
Geographic Coordinates
Range(km) Set Axis Sondrestrom: Electron Temp. Delete Electron Temp
Range(km) Set Axis EISCAT: Electron Temp. Delete Electron Temp
Range(km) Set Axis Millstone: Electron Temp. Delete Electron Temp
Range(km) Set Axis EISCAT: TING model Delete Electron Temp
270 deg
90 deg
UT time
© 1997, University of Michigan

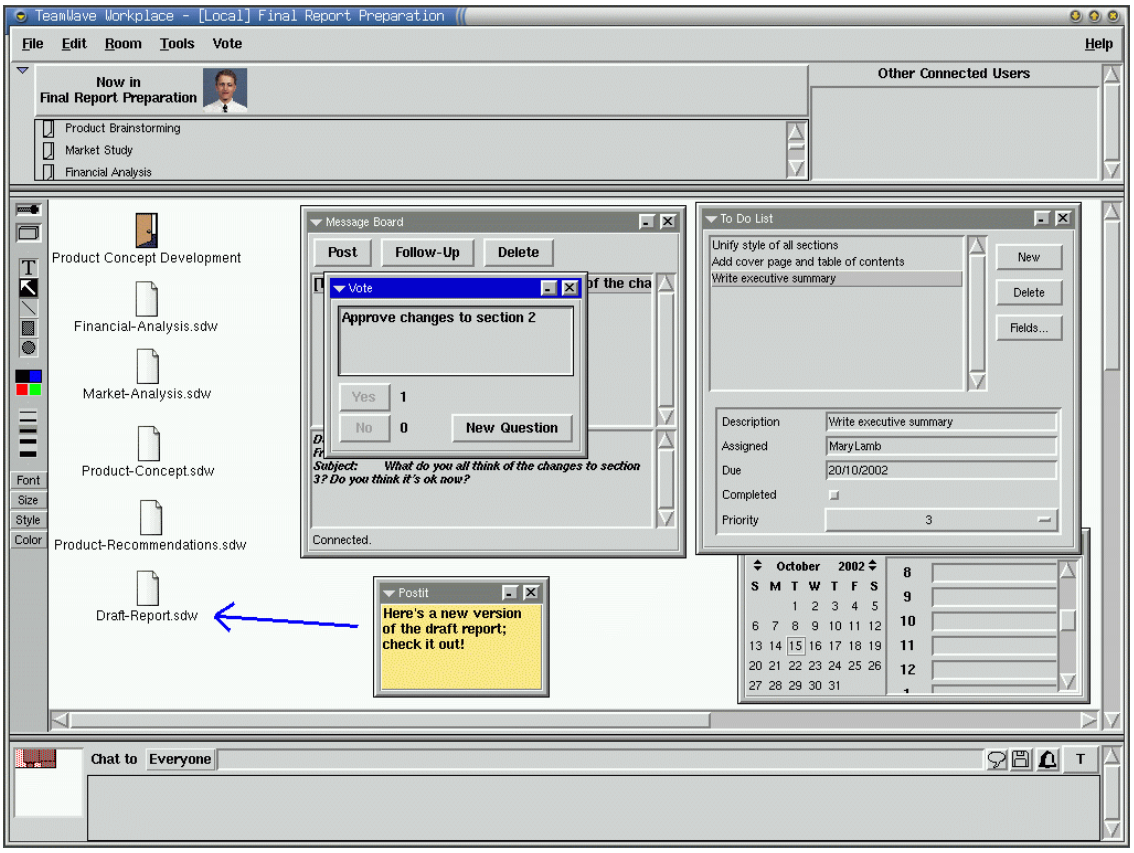
Figure 2.10: CBE generic and domain-specific applets (figure used with permission, courtesy of Hyong Sop Shim and Atul Prakash, Department of Electrical Engineering and Computer Science, University of Michigan)

Access Control

CBE provides four predefined roles (administrator, member, observer, restricted), which users may occupy. Associated with each room is an access control list which stores access privileges for each role. In addition, access control can be enhanced using strong, cryptographic security.

2.2.3.3 TEAMROOMS

The TEAMROOMS collaboration system was developed at the University of Calgary's Grouplab in the mid-1990s (Roseman and Greenberg, 1996). It was subsequently commercialized under the name TEAMWAVE WORKPLACE in 1996. The inspiration for the system came from the observation of collaborating face-to-face business teams, who come together and collaborate in team rooms that are furnished with tools and documents required for their joint tasks. The TEAMROOMS system aims to provide the virtual



Approve changes to section 2

Yes 1
No 0

Subject: What do you all think of the changes to section 3? Do you think it's ok now?

Unify style of all sections
Add cover page and table of contents
Write executive summary

Description	Write executive summary
Assigned	MaryLamb
Due	20/10/2002
Completed	
Priority	3

Here's a new version of the draft report; check it out!

October						2002
S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Figure 2.11: TEAMWAVE WORKPLACE user interface

equivalent of such team rooms to support the collaboration of virtual teams. It is place-based, providing features characteristic of physical places: being long-lived, persistent, providing a venue for communication, offering tools for collaboration and the possibility of bringing in special-purpose tools to customize the place. TEAMROOMS supports both synchronous and asynchronous collaboration. A multitude of tools (termed *applets*) serve as conversational props, such as shared whiteboards or noteboards, and other generic or special-purpose tools for collaboration. TEAMROOMS supports both individual and group work through the use of private and team rooms. Figure 2.11 shows the user interface of one room which is filled with a number of tools (the figure is of the TEAMWAVE WORKPLACE commercial version of the system).

Structuring Metaphor

The collaboration spaces provided by TEAMROOMS are referred to as *rooms*. Upon installation, one room, the "Foyer", is provided as a starting point, but other rooms can be created at any time. Rooms cannot be nested, thus there is no hierarchy among the collection of rooms in TEAMROOMS. *Doorways* can be placed in rooms and configured to connect to other rooms—by clicking on a doorway, its associated room is entered. Thus TEAMROOMS aims to mimic certain of the spatial features of physical team rooms.

Awareness Support

Awareness is provided through several means. In synchronous mode, displays of other users in the same room, together with their idle times, give information on "who is around" and how active they are. This is complemented by several "awareness widgets", such as tele-pointers (other users' pointers appearing on one's copy of the shared space), and radar views (miniature displays of the whole room and the portion in the viewport of different users). Moreover, a complete version history of each object in the shared space is maintained, allowing participants in asynchronous collaboration to trace the changes that have occurred in a room since they last left it.

Access Control

Access control in TEAMROOMS is room-based. Permissions such as entering a room, adding tools, drawing on the whiteboard, etc. can be assigned to just the room owner, or to everyone. Assigning certain access privileges to just some users and not to others, however, is not possible. Thus access control is rather coarse-grained.

2.2.3.4 CVW

The Collaborative Virtual Workspace (CVW) was developed at Mitre Corporation in the mid-1990s (Spellman et al., 1997) to overcome the limitations of other collaboration tools which were seen as being either session-centric, such as video-conferencing tools, or document-centric, such as document management/workflow system. It has since been released into the public domain, where its development continues. CVW's approach is to be *place-based*, meaning that persistent collaboration spaces are provided within which applications, documents and people exist. It supports both synchronous and asynchronous collaboration through a set of generic collaboration services. Figure 2.12 shows a collection of tools in the CVW client's user interface.

Structuring Metaphor

CVW provides *rooms* as collaboration spaces. These are arranged on floors and organized into buildings. Despite this strongly space-based terminology, however, rooms are

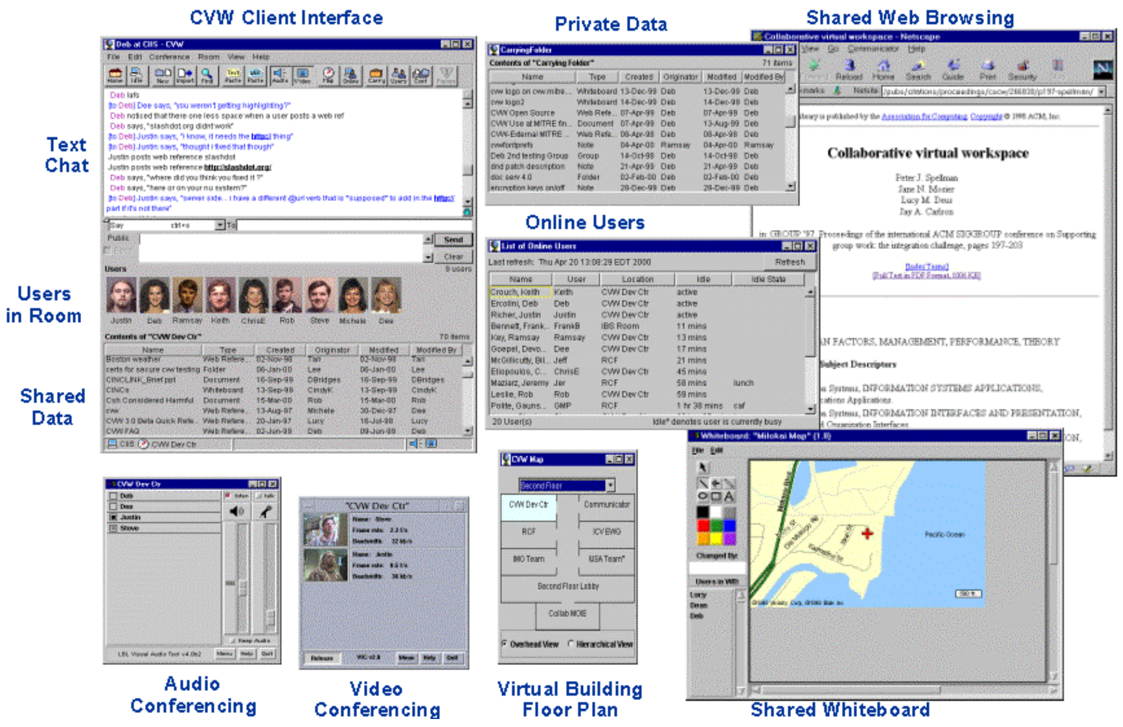


Figure 2.12: CVW user interface (Figure is in the public domain, courtesy of the Mitre Corporation)

presented to the user primarily in the form of abstract information spaces. Only "virtual building floor plans" relate different rooms to each other spatially.

Awareness Support

CVW provides only rudimentary facilities for gaining awareness of others. The main one is a list of online users with idle times, that is an indication of how long ago those users have interacted with the system. This provides basic awareness of "who is around" and how recently they have been active. In addition, the interface for each room similarly has a list of users who are online in the room at that time.

Access Control

Access control is on a room level: access to a room can be restricted through an access control list (ACL) attached to the room. In this case, only users listed in the ACL are permitted to enter the room. Rooms without an ACL are public meaning that everyone is permitted to enter them. Once inside a room, all its contents are accessible. To modify or delete an object, however, requires special rights: usually only the object's owner (or multiple owners) has this privilege, but it can also be assigned to others. Per-object privileges, however, do not exist, thus the access control in CVW is rather coarse-grained.

2.2.3.5 ORBIT

The ORBIT system was developed at the University of Queensland and the CRC for Distributed Systems Technology (DSTC) in 1996, and has since evolved over a number of years (Mansfield et al., 1997; Mansfield et al., 1999). Its design is based on a theory of collaborative activity, Fitzpatrick's locales framework (Fitzpatrick, 1998). The term *locale* is understood to refer to a *place* in the sense of (Harrison and Dourish, 1996), and as discussed above (cf. p. 33). ORBIT is meant to "provide a 'ubiquitous collaborative desktop' through which users will perform all shared and individual tasks" (Mansfield et al., 1997). Users can share access to documents and various objects, and can participate in multiple distinct activities at the same time with different degrees of intensity. Although ORBIT is intended for both synchronous and asynchronous collaboration, it actually only provides support for synchronous communication, through audio-video links and text-based chat. Figure 2.13 shows the user interface of the ORBIT-GOLD client program.

Structuring Metaphor

The main structure in ORBIT is the *locale*, which constitutes an abstract information space populated by people and furnished with documents and objects of relevance to the activities of those people. The developers of ORBIT have purposely chosen this place-based approach instead of utilizing spatial metaphors as others have done, seeing the notion of place as resulting "in greater power and flexibility in CSCW environments" (Mansfield et al., 1997). Locales are seen as places that afford interactions of social groups. As different individuals may be members of different social groups, they consequently may be members of multiple locales. This results in overlapping locales, where some objects or members that exist in one locale can also be found in another locale. To connect different locales whose members need to share objects, so-called *courtyards* can be created. Objects placed in a courtyard are then accessible from all connected locales.

Awareness Support

ORBIT provides a number of facilities for providing awareness. It displays which users are in which locales, and the actions performed by these users on objects in the locales. Awareness is implemented through an event-based notification service which allows users to choose the kinds of events they are interested in receiving. Thus awareness is user-tailorable.

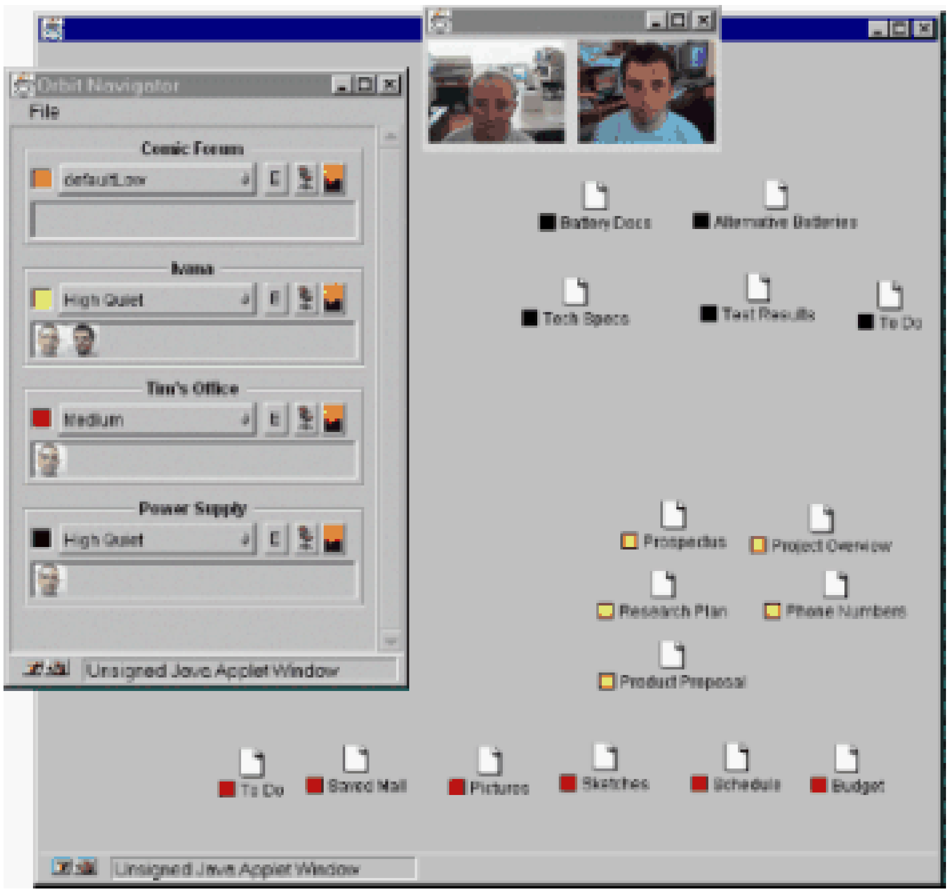


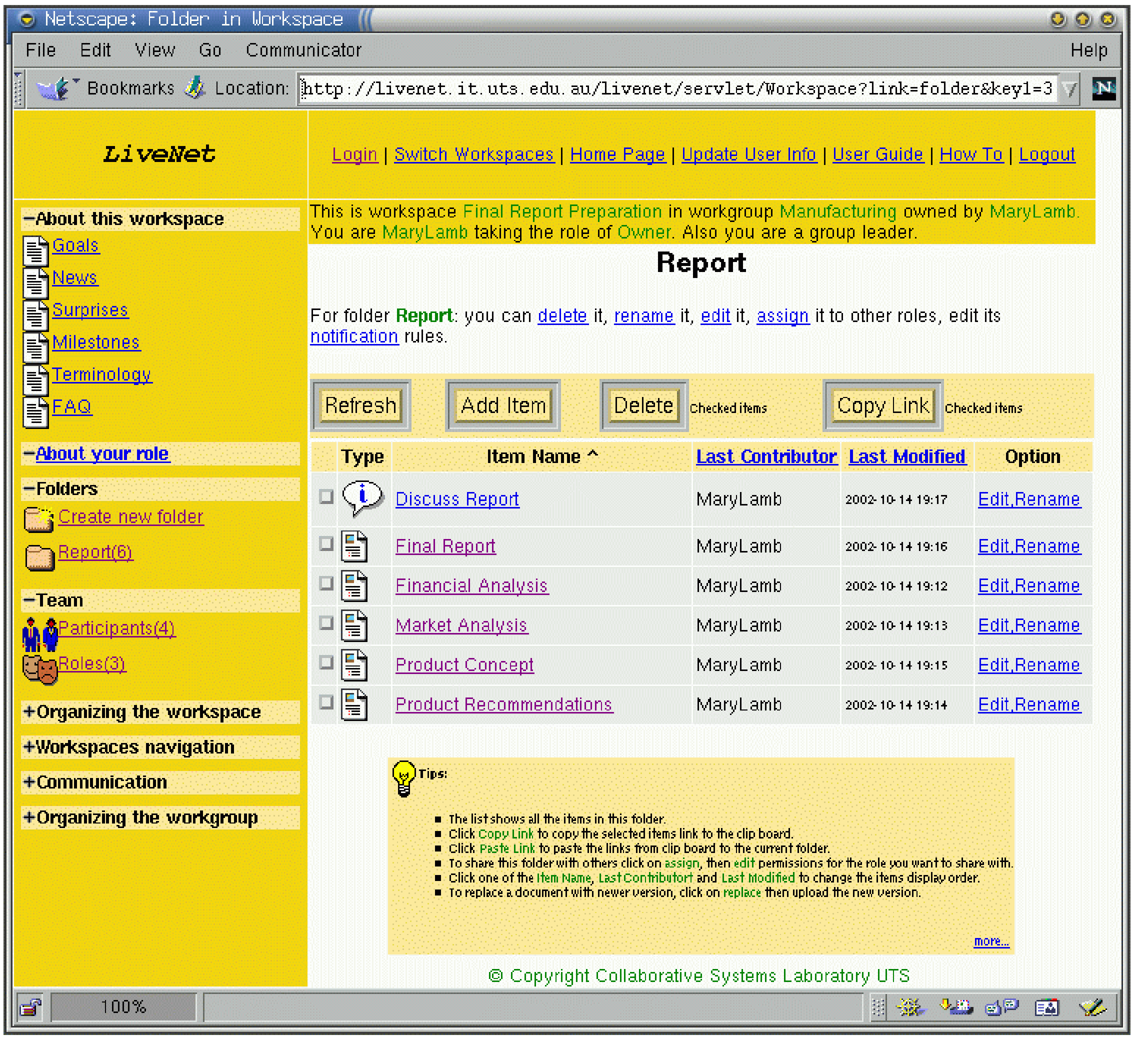
Figure 2.13: ORBIT-GOLD user interface (Figure used with permission, courtesy of Tim Mansfield, Distributed Systems Technology Centre, Sydney, Australia)

Access Control

Access control in ORBIT is locale-based. All members of a locale have full access to all objects within the locale, while all non-members have no access at all to any of the objects. No distinction of types of locale membership or roles exists, and this is perceived to be a shortcoming to be addressed (Mansfield et al., 1997).

2.2.3.6 LIVENET

The LIVENET collaboration system was developed at the University of Technology, Sydney's Collaborative Systems Laboratory in the late 1990's (Hawryszkiewycz, 1999b), and is still being actively developed further. It supports mainly asynchronous collaboration of distributed groups of people, i.e. different-time, different-place interactions, while support for synchronous interactions is being added to the current version of the system. A central server is accessed across the network through one of several client interfaces,


Netscape: Folder in Workspace
File Edit View Go Communicator Help
Bookmarks Location: http://livenet.it.uts.edu.au/servlet/Workspace?link=folder&key1=3
LiveNet
Login | Switch Workspaces | Home Page | Update User Info | User Guide | How To | Logout
−About this workspace
Goals
News
Surprises
Milestones
Terminology
FAQ
−About your role
−Folders
Create new folder
Report(6)
−Team
Participants(4)
Roles(3)
+Organizing the workspace
+Workspaces navigation
+Communication
+Organizing the workgroup
Refresh
Add Item
Delete
Checked items
Copy Link
Checked items
100%

This is workspace Final Report Preparation in workgroup Manufacturing owned by MaryLamb. You are MaryLamb taking the role of Owner. Also you are a group leader.

Report

For folder **Report**: you can delete it, rename it, edit it, assign it to other roles, edit its notification rules.

	Type	Item Name ^	Last Contributor	Last Modified	Option
☐		Discuss Report	MaryLamb	2002-10-14 19:17	Edit,Rename
☐		Final Report	MaryLamb	2002-10-14 19:16	Edit,Rename
☐		Financial Analysis	MaryLamb	2002-10-14 19:12	Edit,Rename
☐		Market Analysis	MaryLamb	2002-10-14 19:13	Edit,Rename
☐		Product Concept	MaryLamb	2002-10-14 19:15	Edit,Rename
☐		Product Recommendations	MaryLamb	2002-10-14 19:14	Edit,Rename

Tips:

- The list shows all the items in this folder.
- Click Copy Link to copy the selected items link to the clip board.
- Click Paste Link to paste the links from clip board to the current folder.
- To share this folder with others click on assign, then edit permissions for the role you want to share with.
- Click one of the Item Name, Last Contributort and Last Modified to change the items display order.
- To replace a document with newer version, click on replace then upload the new version.

more...

© Copyright Collaborative Systems Laboratory UTS

Figure 2.14: LIVENET user interface (web interface)

most commonly through a Web interface (an example of which is shown in Figure 2.14).

LIVENET provides collaboration spaces called *workspaces*. These can be populated with a number of different types of objects such as documents, discussion forums, tools, and message channels. Across workspaces, most of these objects can be shared. For each workspace, any number of *roles* can be defined, and users can be added to a workspace as *participants* occupying roles. The same user can have different roles in different workspaces, but can only occupy one role in each workspace.

For documents in a LIVENET workspace, two types of documents are distinguished: on the one hand documents which form the object of the work performed in the workspace (simply termed *documents*); and on the other hand documents which constitute background material needed for the task (termed *backgrounds*).

Discussion forums constitute the main means of communication. Internally, they are structured as a *forum* (one per workspace), which can have any number of discussion *blocks* (named with the topic of discussion), each of which in turn can contain any number of discussion *statements*. Since statements may be replies to other statements, the collection of statements is a threaded discussion structure like that of many popular bulletin boards, such as Usenet news.

Message channels constitute another means of communication. They are intended for the somewhat more formal communication requirements, where defined lines of communication are known in advance. As message channels are uni-directional they are mostly useful for notification purposes only, such as from the performer of one task to the performer(s) of another dependent task. Messages transmitted over message channels have a pre-defined type, for example "proposal completed", and a semi-structured message body. Internally, a message channel is defined as a *message-rule* specifying source and destination, as well as message type. Message types are separately defined, as are the actual messages transmitted.

Finally, a facility is provided for sending electronic mail from within a workspace to an external email address.

Structuring Metaphor

The structuring metaphor employed in LiveNet is an abstract one: the collaboration space is referred to as a *workspace*, and the collection of workspaces is referred to as a *workspace network*. All workspaces form a tree hierarchy with parent workspaces and subworkspaces, where each workspace has exactly one parent workspace, except for the root of the tree which is parent-less.

Workspaces are grouped together into *workgroups*. Each workgroup has its own tree of workspaces, and therefore its own root workspace. Besides workspaces, a workgroup also has a collection of users. Only users who are members of a workgroup can be added as participants to workspaces in that workgroup. In this way, workspaces associated with different groups of users can be kept separate from other groups. However, the same user may be a member of multiple workgroups.

Awareness Support

Awareness facilities in LiveNet can be divided into two categories. On the one hand, *notification* of certain defined events is provided, which is sent by ordinary email to the user's mailbox, i.e. outside the system. The set of events that can trigger a notification can be configured and includes such events as: uploading of a document to a folder, addition of a new user to a workspace, posting of a statement in a discussion forum, etc. The other awareness facility in LIVENET is rather unique when compared to other collaboration systems, and consists of a set of pages with general task-related information, including: goals, news, surprises, milestones, terminology, and FAQs (frequently asked questions). These aim to assist the user in gaining awareness of the more general aspects of the overall collaboration.

Access Control

Access to the various facilities provided in LIVENET is controlled by access rules. Two sets of rules are provided: the first set of rules applies to the level of workgroups and defines whether a user has permission to add users to and remove users from a workgroup. Users with these privileges have the status of workgroup *leader*. The second set of rules applies to individual workspaces and defines general permissions, such as creating, modifying and deleting workspaces, adding and removing participants, roles and documents, the use of communication facilities, etc. While the first set of rules applies to individual users directly, the second set of rules applies to roles, and thus indirectly to the users occupying those roles. The access control model in LIVENET is thus a hybrid of user-based and role-based access control.

2.2.3.7 Summary and Comparison

The preceding pages have briefly reviewed six collaboration systems that have emerged from research within the past decade, giving a flavour of some of the variety of the systems in existence. These systems differ in a number of ways, both in terms of their implementation and in their support for virtual collaboration. Table 2.1 presents an overview comparing these collaboration systems in terms of seven attributes: architecture, structuring metaphor, support for synchronous or asynchronous collaboration, communication facilities, document sharing facilities, awareness facilities, and access control.

Architecture

In terms of architecture, three main approaches can be distinguished: pure web-based, web-based Java applets, and client-server. While client-server systems typically provide the richest feature sets and user interfaces, they are the most difficult to deploy as they require installation on each client computer, and typically do not work through firewalls. Therefore, many systems adopt the web-based architecture where no installation of client-side software is required (except for a web browser, which is a standard software on today's office computers). The disadvantage, however, is the much less sophisticated

	BSCW	**CBE**	**TeamRooms/ TeamWave**	**CVW**	**Orbit (Orbit-Gold)**	**LiveNet**
Architecture	Web-based (optionally with Java applets)	Web-based Java applets	Client-server	Client-server	Web-based Java applets	Web-based
Structuring metaphor	Abstract spaces (“workspaces”) in tree hierarchy, resembling file system	Abstract “rooms”	Abstract “rooms”	Abstract “rooms”	Abstract spaces (“locales”)	Abstract spaces (“workspaces”) in multiple tree hierarchies
Synchronous/ asynchronous support	Mainly asynchronous; synchronous through external tools	Mainly synchronous; asynchronous also possible	Synchronous and asynchronous	Mainly synchronous; asynchronous also possible	Mainly synchronous; limited support for asynchronous	Asynchronous
Communication facilities	Discussion forums, email, others through external tools	Multi-user text chat	Multi-user text chat, notice board	Multi-user text chat, audio & video channels	Audio & video channels	Discussion forums, message channels, email
Document sharing facilities	Sophisticated document sharing, version management, locking, annotations	Limited to applet data sharing	Document sharing, version management, shared whiteboards	Simple document sharing, shared whiteboards	Document sharing	Document sharing
Awareness facilities	Event history, daily event logs	Users in rooms, applets may provide extra awareness	Users in rooms, user idle time, telepointers, radar views, object history	Users in rooms, user idle time	Users in locales, object-related event history	Object-related event notification, task awareness information
Access control	Hybrid user-based & role-based, fine-grained permissions	Role-based, per-room ACLs	User-based, per-room permissions	Coarse-grained, user-based per-room ACLs	Coarse-grained, user-based per-locale permissions	Hybrid user-based & role-based, per workspace & work-group permissions

Table 2.1: Comparative overview of six collaboration systems

user interface, as well as the slower response time since practically all user actions require a request to be sent back to the server. Also, pure web-based systems (i.e. with an HTML-only interface) typically only support asynchronous collaboration well, because of the difficulty of providing synchronous communication facilities through such an interface. A few systems therefore adopt an approach that lies in between the client-server and the pure web-based approach, using web-based Java applets that connect back to the collaboration system's server through their own network connection (such as a separate socket connection, or using Java remote method invocation). This enables them to provide a more sophisticated user interface than pure web-based systems, including synchronous communication tools. It also facilitates deployment to client computers by avoiding having to install the software on those computers—the applets are simply downloaded and started from the web whenever a specific web page is opened. The disadvantages, however, are the longer time required to download the applets, limitations on applet communication back to the server when firewalls need to be crossed, and the poorer stability—applets tend to crash more easily than traditional client-server applications. Thus in terms of architecture there is no ideal solution, with each alternative having its own advantages and disadvantages, and consequently several different approaches may co-exist.

Structuring Metaphor

All the reviewed systems employ a more or less abstract structuring metaphor. Some are completely abstract, such as the BSCW and LIVENET systems, while others draw on the terminology of physical spaces. For example, CBE, TEAMROOMS, and CVW provide "rooms", while ORBIT makes "courtyards" available. In CVW, "virtual building floor plans" show maps of rooms in spatial relation to each other, however the spatial metaphor is not employed elsewhere in the system. Thus on the whole, the collaboration space itself constitutes an abstract information space in all the reviewed systems.

Synchronous/Asynchronous Support

Both synchronous and asynchronous collaboration depend on suitable communication facilities. For instance, audio/video conferencing is suitable for synchronous collaboration, but not for asynchronous collaboration, as it requires all parties to be present at the same time. On the other hand, a discussion forum may be suitable for asynchronous collaboration, but does not support synchronous collaboration very well. In addition, the support of suitable awareness facilities is a further requirement for both modes of collaboration. In the case of synchronous collaboration, awareness of users in the collaboration space is important, while in asynchronous collaboration awareness of the event history of the collaboration space is of importance. The provision of these communication and awareness facilities, however, is largely dependent on the collaboration system's architecture. Consequently, pure web-based systems are limited to providing support for asynchronous collaboration only, while client-server systems and web-based Java applets have the opportunity for providing support for both synchronous and asynchronous collaboration. In the case of web-based systems, synchronous communication facilities can be optionally provided through the use of Java applets or external third-party applications, such as with the BSCW system which supports both of these approaches.

Communication Facilities

Communication facilities provided by collaboration systems can be divided into two main categories, as mentioned above: those supporting synchronous collaboration, and those supporting asynchronous collaboration. Communication facilities for synchronous collaboration typically include text-based chat and audio/video channels. For asynchronous collaboration, they include discussion forums and notice boards, message channels, and email (which consists of facilities for sending of email only).

Document Sharing Facilities

The sharing of documents is important for most collaborative endeavours, regardless of the mode of collaboration, and consequently document sharing facilities are typically provided. In the simplest case, they allow documents to be uploaded to a server and shared with others, while more sophisticated facilities include versioning, locking, and annotation capabilities. Some systems, such as CBE, do not provide document upload functions but instead enable the sharing of their tools' data. Thus a tool may load and display some data, which may then be viewed by the other members of the virtual team. The same principle applies for shared whiteboards where drawings made on the whiteboard by a team member are visible to other members.

Awareness Facilities

As mentioned earlier, awareness facilities provided by collaboration systems differ depending on whether the system aims to support synchronous or asynchronous collaboration. For synchronous collaboration, typically a list (possibly including pictures) of users in a collaboration space is provided. This may be supplemented by information such as the users' idle time to give an indication as to their involvement in the work of that collaboration space. More sophisticated awareness of the activities of others is provided through telepointers and radar views such as by the TEAMROOMS system. For asynchronous collaboration, awareness is usually provided in the form of the history of events that has transpired in the collaboration space, particularly those which affect objects (such as object creation/modification/deletion). In the case of the LIVENET system, additional awareness of general aspects of the collaboration is provided.

Access Control

All collaboration systems provide some form of control over access to collaboration spaces and the objects they contain. In some cases, this control is very fine-grained, allowing particular types of access to individual objects in specific collaboration spaces to be defined separately for each user, such as in the case of the BSCW system. In other cases, the control is much more coarse-grained, allowing either all or no access permissions for specific users, such as in the case of ORBIT; or allowing access to all or no users (besides the collaboration space's owner), such as in the case of TEAMROOMS. The other way in which access control differs among systems is in whether access is assigned to individual users directly, or to roles and thereby indirectly to users occupying those roles. Most systems support either one or the other type of access control, but some systems support a combination of both user-based and role-based access control, such as BSCW and LIVENET.

This comparison has highlighted the differences between different collaboration systems in terms of a number of attributes. At the same time it has become evident that considered collectively, all these systems share certain characteristics: all of them support virtual collaboration; all provide more or less abstract collaboration spaces; all provide communication and document sharing facilities, awareness, and access control. These shared characteristics define collaboration systems as a separate class of software systems, apart from other classes of software systems such as, say, office productivity applications, or middleware software. Chapter 3 revisits this class of software systems, showing how their information can be modeled.

2.3 Organizational Memory

This research is concerned with obtaining and retaining observations of virtual collaboration from collaboration systems. Such observations constitute records of the activities of the organizations where the virtual collaboration takes place. Records of organizational activity can be regarded as being part of an *organizational memory* (OM). This section reviews what organizational memory is and what the issues in the OM field are.

For more than a decade, the concept of organizational memory has received considerable attention in the fields of information systems, CSCW, management science, organization science, and various others. The idea underlying the concept is that organizations, like individuals, can be regarded as having a memory. In the simplest view, this memory is the sum of the memories of the individuals making up the organization. More complex views accommodate other elements, as is discussed later on. The notion implies that the organization as a whole "knows" something, as long as it can obtain that knowledge from within its memory.

Knowledge of past actions and decisions is intended to inform present decision-making, as well as to fuel processes of organizational learning. The latter is understood to consist of information processing which affects organizational behaviour and decision making. As defined by (Huber, 1996, p. 126): "An entity learns if, through its processing of information, the range of its potential behaviors is changed". In this context, organizational memories are seen as critical for organizational learning: only when an organization has the ability to recall that which it has learned and stored in its memory can it use its learning (Huber, 1996, p. 150).

The problem related to organizational memory is that "organizations frequently do not know what they know" (Huber, 1996, p. 149). Organizations are found to be "reinventing the wheel", and "have serious limitations in transferring previous learning to current problems" (Conklin, 1993, pp. 561–2). That is, while the required knowledge may reside somewhere within the organization, such as in the heads of its employees, it may not be accessible to other organizational members and consequently is of little value. Moreover, sharing the knowledge of organizational members may be difficult, particularly when the knowledge is tacit and thus not readily sharable (this is discussed in more detail in Section 2.3.2). When these members leave the organization, part of the overall organizational memory is lost.

This case of "organizational amnesia" has motivated researchers in the CSCW and information systems fields to try to provide some sort of computer-based support that can enable organizations to remember aspects of their past. Thus the *notion* of organizational memory is translated into *systems* for organizational memory. Such systems should continually be updated with records of the decision stimuli and responses that occur within the organization, to enable later "remembering" of what the organization knows. In this vein, in Walsh and Ungson's seminal paper on the issue, the behaviour of organizations is characterized in terms of information processing activities of acquisition, retention, and retrieval (Walsh and Ungson, 1991). To Walsh and Ungson, "...organizational memory refers to stored information from an organization's history that can be brought to bear on present decisions" (Walsh and Ungson, 1991, p. 61). This view is echoed by Huber: "Organizational memory is the means by which knowledge is stored for future use" (Huber, 1996, p. 127). Some views on organizational memory, however, are more narrow in scope, such as the one espoused by Conklin who sees organizational memory as "the record of an organization that is embodied in a set of documents and artifacts" (Conklin, 1993, p. 561). A similar view is expressed by Ackerman to whom organizational memory is "organizational knowledge with persistence" (Ackerman, 1994). These artefact-oriented views stand in contrast to other more process-oriented ones such as the following: "Organizational memory (OM) is a generic concept used to describe the acquisition, retention, maintenance, search, and retrieval of knowledge within an organization" (Corbett et al., 1999). By and large, however, these different views agree that an organizational memory supports the remembering of aspects of the organization's past, which exist in one form or another as pieces of information or knowledge.

2.3.1 Data, Information, Knowledge

Here, the meaning associated with the terms "data", "information", and "knowledge" is clarified. These terms are frequently used with different meanings in different situations by different authors. The definition of some of these terms is subject to much controversy, such as the distinction between information and knowledge; a few attempts to review and/or define the meaning of these terms in the context of information systems are provided in (Tuomi, 1999; Spiegler, 2000; Nunamaker Jr. et al., 2001). An important notion associated with these terms is that they constitute a *hierarchy of increasing value* (Ackoff, 1996). That is, the value associated with information is higher than that of data, and likewise the value of knowledge is higher than that of information. Value in this context refers to potential usefulness—the more value something has, the more potential it has to be useful (Nunamaker Jr. et al., 2001). Usefulness in turn is typically seen in the extent to which something can influence and/or guide action. Thus knowledge, which in this hierarchy has the highest level of potential usefulness, has been defined as "information made actionable" (Vail III, 1999). This view is confirmed by (Spiegler, 2000) who sees information as "knowing-that", i.e. being concerned with facts; while knowledge is seen as "knowing-how", i.e. being concerned with the ability to turn information into action.

The view taken here concurs with that of (Spiegler, 2000): "when we attempt to capture, record or store knowledge it turns back into information or data"; while (Alavi and Leidner, 1999) suggest that "information becomes knowledge once it is processed in the mind of an individual". Thus that which is referred to as "knowledge" can only ever reside in the minds of individuals. For the purpose of this thesis, the terms data, information, and knowledge are defined as follows:

> **Definition 12** ***Data*** *refers to uninterpreted raw facts.* ***Information*** *is interpreted data, such that it is given meaning.* ***Knowledge*** *is information made actionable as a result of cognitive effort.*
> □

To illustrate this, "152", "smith.pdf", and "2002/06/22" are examples of items of data. Without interpretation they do not convey meaning in and of themselves (although of course the meaning can in some cases be *guessed*, but this then already constitutes an interpretation, perhaps based on previous encounters with the same or similar data). Data such as the one above is transformed into information by interpreting it and associating meaning with it, such as by saying "paper submission number 152 has the name smith.pdf and was created on June 22nd, 2002". Each of the data items now has become meaningful; moreover, relationships between the three pieces of data have now become evident: they all refer to the same instance of a paper submission. Processing this information and relating it to other information, knowledge can be produced. For instance, relating it to the information "papers will be accepted until June 15th, 2002", the knowledge "paper 152 should be rejected because it was submitted late" can be produced. In this way, the original facts have become actionable knowledge.

2.3.2 Knowledge Creation

The knowledge which is the object of organizational memory can be divided into two categories: *explicit* knowledge and *tacit* knowledge.

Explicit knowledge (as conceived of in the literature) has the property that it can be codified, for example by writing it down or recording it in a form that enables its sharing with others. Thus explicit knowledge can be transmitted between people, possibly through an intermediary such as a computing system.

Tacit knowledge, on the other hand, resides in people's minds and finds expression through their actions and decisions. It is closely associated with experience and learning.

While explicit knowledge is relatively easily captured, stored and retrieved in an information system, tacit knowledge first needs to be converted to explicit knowledge before it can be thus handled. Nonaka has developed a theory that explains how this conversion of knowledge takes place in organizations (Nonaka, 1994). According to this theory,

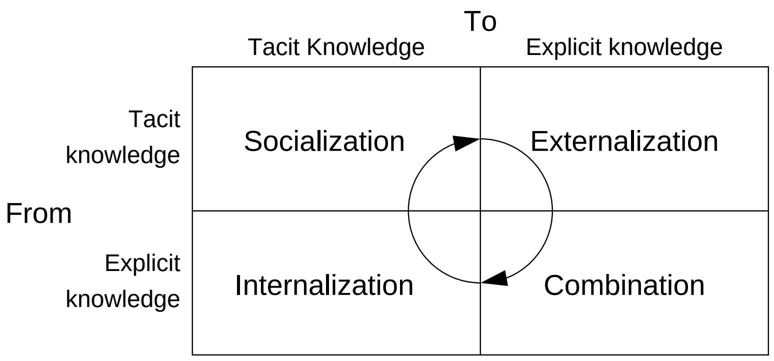


Figure 2.15: Modes of knowledge conversion and the cycle of knowledge creation; adapted from (Nonaka, 1994)

knowledge is created in iterations of four consecutive steps that he terms socialization, externalization, combination, and internalization.

Socialization occurs when people share experiences with each other. The on-the-job training of an apprentice by a master is a typical example of socialization. This creates tacit knowledge in one person from the tacit knowledge in another person by enacting it. Externalization happens when tacit knowledge is articulated and brought into an explicit form. Combination creates knowledge by integrating existing information and knowledge in new meaningful ways. Thus, explicit knowledge is created from other explicit knowledge. Finally, internalization takes place when people apply explicit knowledge through "learning by doing", thus creating tacit knowledge. When performed iteratively, a cycle of knowledge creation results, what Nonaka calls the *spiral of organizational knowledge creation*. The four modes of knowledge conversion and the cycle of knowledge creation are shown in Figure 2.15.

2.3.3 Locus of Organizational Memory

As an organizational memory contains the knowledge an organization possesses, the question arises where this knowledge resides, i.e. where the locus of organizational memory is. Walsh and Ungson have proposed a framework of organizational memory that suggests that the memory's retention facilities are structured in terms of five *retention bins* (Walsh and Ungson, 1991). These retention bins are:

1. *Individuals*, the members of an organization whose own memories contain a record of what has transpired in the organization. Individuals use records and files as memory aids.

2. *Culture*, a “learned way of perceiving, thinking, and feeling about problems that is transmitted to members in the organization” (Walsh and Ungson, 1991). Culture influences decision-making in an organization and is mostly passed along orally.

3. *Transformations*, which take inputs and produce outputs, be they material or intangible, embody a logic that can be regarded part of organizational memory.

4. *Structures*, expressed in terms of the roles that members of the organization occupy, have certain expected behaviours attached. However, it also extends to organizational structure, which contains information on how the organization views its environment.

5. *Ecology*, referring to the physical structure of the organization and its workplace arrangement, holds information about the organization and its members.

In addition to these five retention bins that are internal to the organization, a sixth one is identified: *external archives*, including such things as data compiled by competitors or government bodies, information held by former employees, stories prepared by the news media, etc.

Walsh and Ungson’s framework is comprehensive, and is often quoted in the OM literature. However, it is not without its critics. Bannon and Kuutti comment:

> “The conceptual framework that is proposed by Walsh & Ungson is comprehensive, but it suffers from an attempt to include virtually everything, so that one is left wondering what, within organizations, is not a part of organizational memory?” (Bannon and Kuutti, 1996)

This criticism seems only partly justified. While it is true that the framework is complex, so is the reality it attempts to model. The problem seems to be not related to the concept of organizational memory itself, but rather to the way it can be harnessed to inform decision-making in the organization. Furthermore, while current OM systems only support a small part of the above framework, this should be seen as a motivating challenge to explore if and how information relating to other parts of the framework, that have thus far been elusive, could possibly be obtained and made part of the organizational memory. Chapter 4 again touches upon this question.

2.3.4 Levels of Organizational Memory

Organizations are collections of people, and arguably the most important repositories of organizational memory are the minds of its people. However, organizations are also

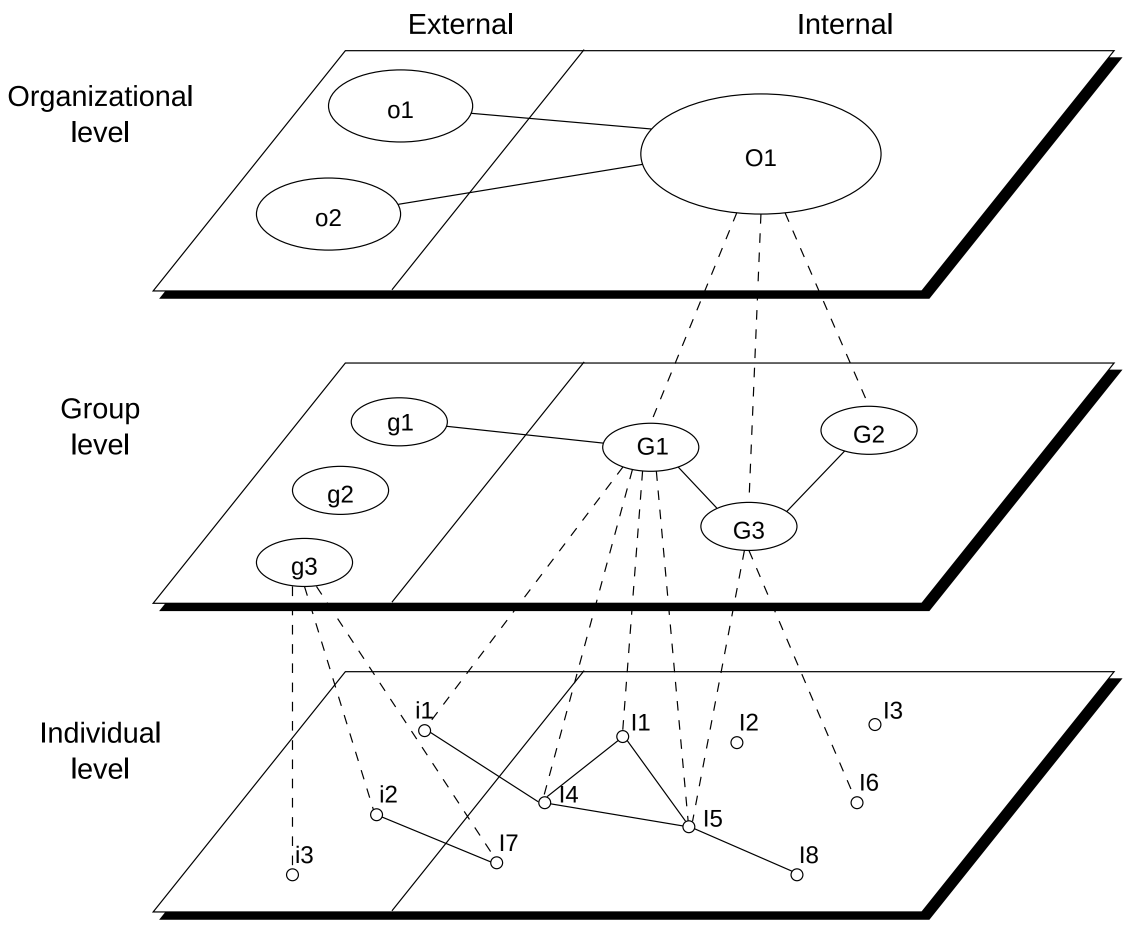


Figure 2.16: Multiple levels of organizational memory; adapted from (Anand et al., 1998)

internally organized in smaller units, such as groups. When considering the different groupings of people in different organizational units, the scope of the memory available in those groupings can be identified. One model that attempts to represent levels of organizational memory has been proposed in (Anand et al., 1998). It identifies the three levels of *individual*, *group*, and *organization*. It then incorporates the concept of *transactive memory* to define the scope of organizational memory at each level. Transactive memory is that portion of OM which does not reside in a particular individual, but which that individual knows how to obtain from another individual, through a set of communication transactions.

Because of the notion of transactive memory, an organization's memory can actually include portions that lie outside the organization, namely through communication of a member of the organization with an outside individual. Likewise, this communication can involve entire outside groups and organizations. A simplified representation of this arrangement is shown in Figure 2.16 (the original representation chosen by Anand *et* *al.* differentiates between access to only explicit, or both tacit and explicit knowledge; these details have been omitted here for the sake of simplicity). The figure shows both internal and external individuals and units. It shows that organizations are composed of groups, which in turn are composed of individuals (composition is indicated by the dashed lines). Thus an organization's memory is the sum of the memories of its contained groups, and in the same manner, the group's memory is the sum of the memories of its contained individuals. It can also be seen that groups internal to the organization may include external individuals, and vice versa. The solid lines among pairs of individuals, groups, and organizations indicate access to the others' knowledge. Thus individual I5, for example, is able to access knowledge of individual I8, by means of a communication transaction.

OM systems have the potential to eliminate the need for some of the communication transactions among organizational members. This effectively broadens the scope of availability of organizational memory by making it available even where no access to the knowledge of others would have otherwise existed.

2.3.5 Declarative vs. Procedural Memory

Most discussions of organizational memory assume that its contents are records of what an organization has done, i.e. decision stimuli and responses. However, this constitutes only one kind of organizational memory, usually termed *declarative memory*. Another kind of organizational memory, however, relates not to *what* has been done, but to *how* something was done or should be done, and this is termed *procedural memory* (Moorman and Miner, 1998).

The term *procedural memory* describes a broad category of organizational memory. In terms of Walsh and Ungson's framework, it encompasses elements of *individuals* (as in "the way I do things") and *culture* (as in "the way things are done around here"), but particularly as *transformations*. As prescriptions for work, such transformations are expressed as procedures, rules, and formalized systems (Walsh and Ungson, 1991, p. 65).

Early efforts at capturing organizational memory focused primarily on declarative memory, such as the work on Answer Garden (Ackerman, 1994) and its successor Answer Garden 2 (Ackerman and McDonald, 1996), both of which facilitate the interactive evolution of a body of declarative knowledge. On the other hand, the need for retaining procedural aspects of work was already pointed out relatively early on (Conklin, 1993). However, the cost of capture was seen as too high:

> "The most immediate barrier to capturing more of the process of work and making it part of organizational memory is that it seems to present an in-

> surmountable and onerous documentation burden on the people doing the work.” (Conklin, 1993)

This is because it was expected that people carrying out the work would themselves contribute records of what they were doing to the organizational memory. As doing so usually benefited others more than the people contributing these records, there was very little incentive to motivate such contributions. Also, making records of one’s work interfered with the work itself, an undesirable situation. Conklin suggested that groupware should be linked with organizational memory to ongoingly tap into the flow of interactions between members of an organization, and to crystallize this into the organizational memory. This has been echoed more recently in the context of virtual team effectiveness, where the following “effectiveness dimension” has been proposed:

> “The degree to which the team’s process and outcomes can be captured electronically, stored and retrieved as needed to contribute to increased levels of organizational knowledge and learning for future teams.” (Furst et al., 1999, p. 253)

Despite this support for the notion of capturing and retaining procedural memory, there has to date been very little practical work in that direction. The issue of the capture and retention of procedural memory is revisited in Chapter 4 where it is suggested how such a memory can be created and maintained.

2.3.6 Remembering

In the context of organizational memory, remembering is usually understood as being synonymous with query and retrieval. That is, if an organizational memory contains some material, remembering that material means formulating a query for its retrieval. This view of remembering considers memory as a passive store.

Drawing on early psychology research, Bannon and Kuutti have proposed an alternative view where memory is seen as a constructive act (Bannon and Kuutti, 1996). They argue that:

> “Each action of memorizing or storing information and each action of recalling and remembering take place in the context of an activity. If storing context and recalling context are the same activity, the interpretation of the material may not be problematic. But if remembering takes place in a different activity where material has been stored, the material will be reinterpreted

> with respect to the new object or activity, and there is no automatic guarantee that the material is relevant anymore in the same way than it was in the context of storing it." (Bannon and Kuutti, 1996)

So, an act of remembering is seen not as mere retrieval, but as constructing meaning within a certain context. Removed from that context, some of the meaning of that original material may well be lost. An organizational memory system thus needs to allow for the interpretation and the shared assigning of meaning in the context of remembering if it is to provide any benefit from knowledge of the past in present decision-making.

2.3.7 Forgetting

It has been suggested that an organizational memory system should have the ability to forget (Landry, 1999). This is justified by comparing the OM with human memory. Humans need to forget in order to cope with overwhelming amounts of information, otherwise they suffer information overload, and are bogged down with a deluge of mindless trivia. The danger of not forgetting is the risk of being caught up in the past, without adapting to changed circumstances in the present, resulting in poor decision-making. On the other hand, total amnesia is dangerous too, as it prevents learning from the past. Thus, a "good" memory should combine the right amount of remembering with the right amount of forgetting. The challenge then is "to build systems that are appropriately forgetful" (Landry, 1999).

On the other hand, to compare OM with human memory and then conclude that an OM should mimic human memory in every aspect seems to deny the shortcomings of human memory. For forgetting not only frees us of "mindless trivia", it also suffers us to lose important information, sometimes with disastrous consequences. Landry's claim is that organizational forgetting is overall beneficial, that the instances when forgetting jeopardizes the organization's survival are outweighed by those when remembering leads to poor decision-making.

The obvious alternative to forgetting in the context of OM systems is *filtering*, that is, the selective withholding of material. Landry briefly addresses, and dismisses, this possibility, however without offering any convincing reason why not to utilize it. Filtering appears to be the most suitable solution for dealing with excessive amounts of information. In the literature, however, this has been a relatively unexplored area, which seems deserving of further research.

2.3.8 Organizational Memory vs. Knowledge Management

Before closing this discussion on organizational memory, some words should be said about its relationship to knowledge management (KM), which is often mentioned in the context of, and sometimes seen as synonymous with, organizational memory. Because most research into knowledge management has been relatively recent, its concepts and terminology are still somewhat fluid and at times vague. Thus there is no universally agreed-upon definition of knowledge management, or a clear distinction between it and organizational memory. For instance, (Maier and Lehner, 2000) consider KM systems to be a subset of OM systems; by implication, one should regard KM as being subsumed under OM. However, others take the opposite view, such as (Katzy et al., 2000) who regard the OM to be the knowledge repository of a KM system, thus OM being a subset of KM. Others take a more process-oriented view, such as the following: "Organizational Memory (OM) can be defined as the way an organization applies past knowledge to present activities. Knowledge Management (KM) addresses the process of acquiring, creating, distributing and using knowledge in organizations" (Morrison and Olfman, 1999). This definition of KM is strikingly similar to that of OM by (Corbett et al., 1999) presented earlier. The difference between the two may indeed be a subtle one in terms of definition. However, it appears that by and large the focus of organizational memory is the memory component itself, as a repository of the organization's knowledge; while that of knowledge management is the process of acquiring, representing, and disseminating knowledge within the organization. Thus the distinction can be seen as that between an entity and a process.

2.4 Summary

This chapter has provided an overview of the problem domain of this research work: virtual collaboration processes, collaboration systems, and organizational memory.

Virtual collaboration processes are understood to be goal-directed collections of multiple tasks, involving multiple individuals, and performed without face-to-face interaction. Several different types of such processes can be identified, which can be classified according to a number of process attributes. Common among these classifications is the attribute of predefinition, which can be used to broadly divide between production processes and emergent processes. For modeling these processes, a number of modeling methods and notations are available, and this chapter has reviewed several of them.

In order to carry out virtual collaboration, collaboration systems are employed. These provide collaboration spaces structured according to some abstract or spatial metaphors.

Collaboration spaces provide certain facilities to enable virtual collaboration, including document sharing, communication, and awareness facilities, supporting synchronous and/or asynchronous collaboration. Many such systems have been developed, and a number of them were reviewed in this chapter.

Finally, observations of virtual collaboration can be seen as belonging to an organizational memory. Such a memory contains knowledge of what the organization "knows", in terms of experience from its past which it can bring to bear on present decisions. Several aspects relating to organizational memory were investigated, including its locus, multiple levels, declarative and procedural content, remembering, and forgetting.

Having separately reviewed these areas in this chapter, the following chapters make the connection between this research and these areas of the problem domain, relating and integrating these areas.

Chapter 3

Modeling Patterns of Virtual Collaboration

As stated in the previous chapters, this thesis is concerned with obtaining observations of virtual collaboration. Since the observations are to be collected without requiring members of virtual teams to document their own actions (and also without requiring others to do so), the collaboration systems used by the virtual teams are the only practical information sources. Thus, a given collaboration system needs to collect records from which such observations can be made. The challenge, however, is that the records typically collected by such systems on the one hand, and the observations that are sought on the other hand, usually differ significantly in both the amount of detail contained and the scale of the activity represented. To illustrate this, consider the records shown in Figure 3.1 which originate from one particular collaboration system. The observations sought, on the other hand, are more of the kind as shown in Figure 3.2 (cf. the discussion on MOO diagrams on page 29 for a reminder of the notation). Here, the records in Figure 3.1 correspond to only a small part of the overall task represented in Figure 3.2. In terms of the amount of detail, the records in Figure 3.1 contain a large amount of detail, while the representation of the collaborative activity sought, as shown in Figure 3.2, contains much less detail. In terms of the scale of the activity represented, the records from the collaboration system shown in Figure 3.1 are of a very small scale, representing actions performed by a single user in a given collaboration system; while the corresponding representation of the collaborative activity shown in Figure 3.2 is of a much larger scale, representing an aggregation of multiple actions performed by multiple users. To bridge from the former to the latter is the object of this and the following chapter. As will be argued in these two chapters, doing so involves the combination of two things: the modeling of information about virtual collaboration, and the derivation of information about virtual collaboration.

```
161.64.61.126 - - [25/Aug/2000:09:40:55 +1000] "GET http://livenet.i
t.uts.edu.au/livenet/servlet/Document?key1=1367667488&key2=-18563551
85&folderName=Report&document=http://livenet.it.uts.edu.au/docs/Repo
rt/FinalReport.pdf" 200 1980
161.64.61.126 - - [25/Aug/2000:09:41:51 +1000] "GET http://livenet.i
t.uts.edu.au/livenet/servlet/Discussion?link=discussionOpen&key1=136
7667488&key2=-1856355185&discussion=Discuss-Report|disc://Report-Pre
paration.John_Smith.Report.Discuss-Report&folderName=Report" 200 653
```

Figure 3.1: Records from a collaboration system

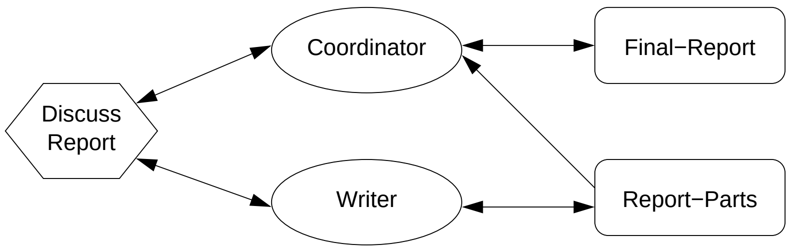


Figure 3.2: MOO diagram of a report preparation task

The current chapter deals with the former, the modeling and representation of information about collaboration. The following chapter then carries on from there to deal with the latter, the method for deriving the observations sought from a collaboration system's records.

3.1 Patterns

It is proposed that observations of virtual collaboration should be regarded as *patterns* that can be modeled, and then extracted, from data on collaboration. The MOO diagram in Figure 3.2 is an example of a simple pattern of activity between two individuals. At this point, some words are in order about what is meant by the term "pattern".

Within the computing sciences, the term stands associated mainly with following two concepts: on the one hand, a pattern is understood to be a structure existing in a body of data (Fayyad et al., 1996); this is essentially the meaning which the term occupies in the areas of data mining and pattern recognition. On the other hand, a pattern is understood to be the specific combination of classes and objects for solving a certain type of design problem (Gamma et al., 1995); this is the meaning which the term occupies in the area of object-oriented software construction.

The main distinction between these two concepts is this: in the former case, i.e. a pattern as structure in a body of data, a pattern is understood to be *descriptive* in that it conveys something about a pre-existing body of data. In the latter case, however, i.e. a pattern as a combination of classes and objects, a pattern is seen as being *prescriptive* in that it expresses how software should be constructed. A further distinction between these two meanings of the term is that for the former, the descriptive pattern, the *source* of the pattern is the body of data itself. That is, the pattern already pre-exists within the data and is subsequently discovered. The latter, the prescriptive pattern, however, does not exist of its own, but rather is the product of a human cognitive process, i.e. its source is the tacit knowledge of a human expert, in this case a software designer.

The idea of patterns has been applied to the broad domain of collaboration by a number of researchers. For patterns in the prescriptive sense, some of the earliest work has been by Coplien and his colleagues who have investigated the software development process, compiling a collection of patterns of productive software organizations (Coplien, 1995; Harrison and Coplien, 1996). Within the domain of online learning, Wessner and Pfister have suggested the concept of *Points of Cooperation*, which somehow seem suggestive of patterns for cooperative interactions (Wessner and Pfister, 2000). Briggs and his colleagues have created the notion of *thinkLets* as patterns of group facilitation within the domain of Group Support Systems (GSS) (Briggs et al., 2001). In the domain of workflows, the notion of *workflow patterns* as basic building blocks of workflows has recently been proposed (van der Aalst et al., 2003). Finally, IBM has developed a set of *e-business patterns*—architectural patterns used in the construction of e-business systems, including those facilitating collaboration—together with a methodology for applying them (Adams et al., 2001).

For patterns in the descriptive sense, on the other hand, Erickson has suggested the use of pattern languages for making the results of workplace studies more easily reusable (Erickson, 2000). In a similar vein, Martin and his colleagues at Lancaster University have suggested the use of patterns of cooperative interaction to inform design, where the patterns are drawn from ethnographic studies of work environments (Martin et al., 2001).

The patterns of virtual collaboration that are sought here are patterns in the descriptive sense, i.e. structures within data. Structure within data refers to the *interrelation* and *arrangement* of individual units of data within a larger body of data; i.e. related units of data placed together in a specific arrangement. A database record is an example of structure within data, as it constitutes an arrangement of multiple interrelated fields (where each field typically represents an attribute of some entity, such as in a record consisting of the name and address of a person).

Individual structures may be combined to form larger structures. The example of the records from a collaboration system shown in Figure 3.1 is a case in point: the two records are units of data that are interrelated (recording actions performed by the same user, in the same collaboration space) and arranged in consecutive order. Together these are part of a larger structure, which is represented in an abstract form as the MOO diagram of Figure 3.2. This abstract description is a pattern:

> **Definition 13** *A **pattern** is an abstract description of the structure of a body of data.*
> □

A pattern of virtual collaboration is therefore an abstract description of the structure of a body of data related to virtual collaboration. It relates to users, collaboration spaces, artefacts, etc., and to certain relationships among these. In order to get from a body of data to a pattern, however, it is necessary to model the information of each one. The following section deals with this issue in more detail.

3.2 The Information Pyramid of Virtual Collaboration

Patterns of virtual collaboration are obtained from a source body of data about virtual collaboration. At the beginning of this chapter, the disparity in terms of the amount of detail and the scale of the activity between the two was mentioned: the source data provided by collaboration systems is typically very detailed and represents very small-scale activity, while the patterns sought are of much larger scale activity and much less detailed. Here it is suggested that this disparity is too great to be bridged in a single giant step. Instead, information related to virtual collaboration should be considered at a number of different *levels of abstraction*. Each of these presents a different *view* of the same information. Here, it is proposed to consider six different levels, named *infrastructure*, *system*, *user*, *collaboration*, *task*, and *process* levels, with six different views, as shown in Figure 3.3.

Each of these levels consists of information about virtual collaboration. This information is of two kinds: on the one hand there is information about the entities in collaboration spaces, as well as their combination into specific configurations; this is referred to as *static* information. On the other hand there is information about the actions that take place within a collaboration space; this is referred to as *dynamic* information. Static information represents *structures* of virtual space, while dynamic information represents *behaviour* associated with those structures.

Static information consists of *objects* provided and maintained by the collaboration system:

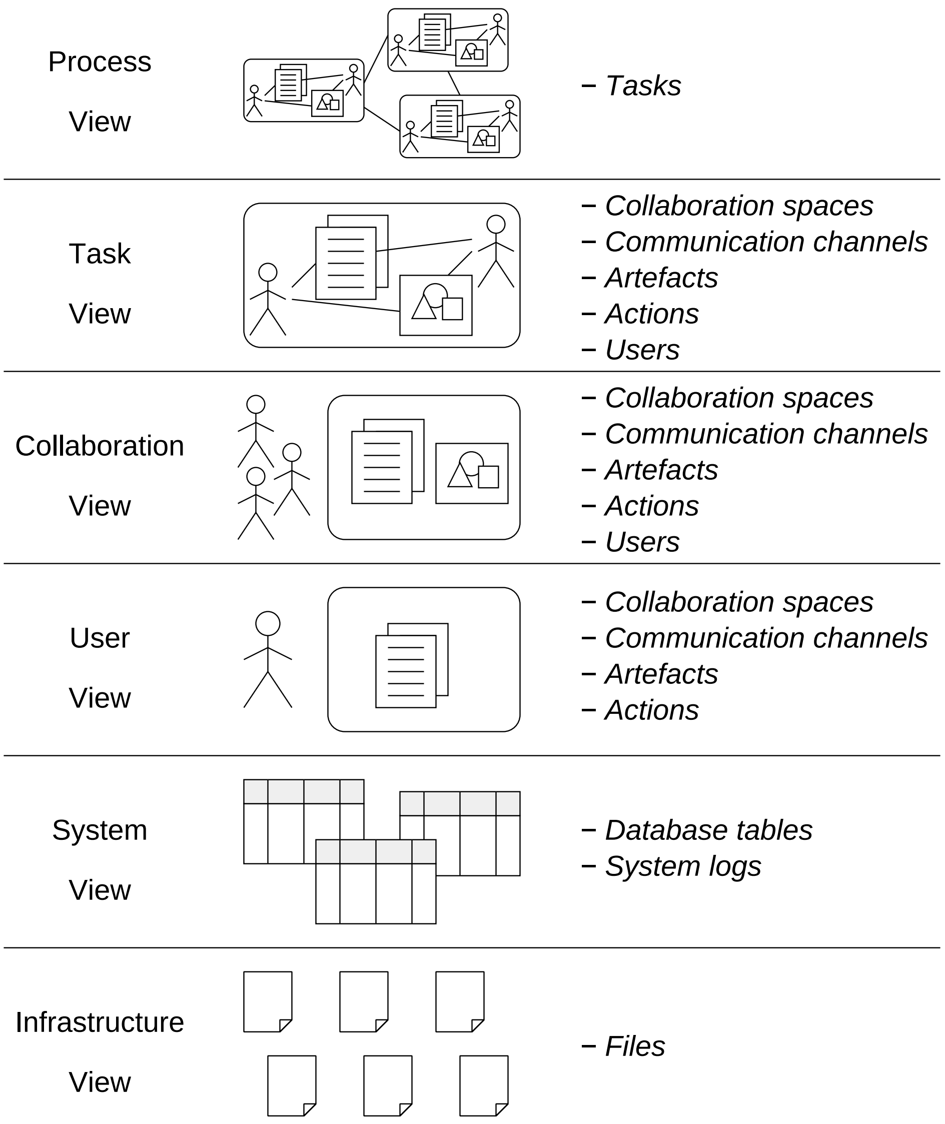


Figure 3.3: Views of information about virtual collaboration at different levels of abstraction (left and centre) and the contents of each view (right)

Definition 14 *An **object** is a static entity provided and maintained by a collaboration system. It consists of one or more* attributes *that describe it. The set of values of an object's attributes at a given point in time constitutes the object's* state *at that time.*
□

Examples of objects are collaboration spaces, documents, discussion forums, users, messages, etc. Attributes are meta-data related to the object. Examples of the attributes of an object, say a document, are its name, creator, creation timestamp, etc.

Dynamic information consists of *actions* that occur within a collaboration system:

Definition 15 *An **action** is a function or operation that can be performed in a collaboration system. It consists of one or more* attributes *that describe it.*
□

Examples of actions are creating a collaboration space, opening a document for reading, posting a statement to a discussion forum, etc. An example of the attributes of an action, say for the action of creating a collaboration space, are the name of the new collaboration space, and the name of the user who is to be the new collaboration space's owner. Actions themselves are *stateless*. However, actions usually affect objects, and may alter the state of one or more objects. Actions are performed by *action performers*: either humans (such as the users of a collaboration system), or software systems (such as the collaboration system itself).

The definition of an action above is *minimal* in that it includes only attributes describing the action itself. However, the action performer, that which is being acted upon, the location of the action, and the time of the action, are not among the attributes describing an action, but rather form the *context* of the action. In order to fully describe an action in any meaningful way, however, requires the inclusion of its context. Action context is defined as follows:

Definition 16 *An **action context** is the set of information identifying the subject, referent, location, and time of an action.*
□

The subject is the action performer, which as mentioned above is either a human user or a software system. Every action context must have at least one subject, but it is possible for an action context to have multiple subjects in cases where more than one subject participates in the performance of an action. The referent is that which is being acted upon, an object such as a collaboration space or discussion forum[1]. Not every action context has a referent, depending on the action it relates to, while some action contexts may have multiple referents. Location is the place where the action occurs, such as a collaboration space. Every action context must have at least one location, but it is possible for an action to take place in multiple locations simultaneously and thus for the action context to identify multiple locations. Finally, time is the instant or period in time when the action takes place, and this information must be present in every action context. Subject, referent and location are objects, while time is a data value (in the case where time is an instant) or a pair of data values (in the case where time is a period), identifying any or all of year, month, day, hour, minute, second, and millisecond.

A given action may occur in many different action contexts. For example, the action of posting a discussion statement could be performed by different subjects (users, roles); have different referents (discussion forums); be performed in different locations (collaboration spaces); and take place at different times. Collections of similar actions can be *generalized* into *action patterns*. Recall Definition 13 of patterns above: “a pattern is an abstract description of the structure of a body of data.” Based on this definition, an action pattern is defined as follows:

> **Definition 17** *An **action pattern** is a pattern describing an action together with a particular action context.*
> □

That is, an action pattern combines an action and an action context. Whereas an action refers to only the actual activity performed, and the action context refers to information related to that activity but not including the activity itself, an action pattern brings these two together.

As an action pattern describes an activity of virtual collaboration, the term “action pattern” is synonymous with the term “pattern of virtual collaboration”: both mean the same and may be used interchangeably.

An example of an action pattern is the one depicted in Figure 3.2 above, consisting of action “Report Preparation” together with the action context consisting of subjects “Coordinator” and “Writer”, referents being a discussion forum named “Discuss Report” and two documents named “Report-Parts” and “Final-Report”, location being a collaboration space “Report Preparation”, and time being the period 20/8/2000–12/9/2000[2].

[1]In grammar this is usually called the *object*, however since this term is already being used here with a different meaning (see Definition 14 above), the term *referent* is used instead in order to avoid confusion.

[2]Note that the MOO diagram in Figure 3.2 does not show the location and time.

In each of the six views of information of Figure 3.3, there are objects, actions, action context, and action patterns. However, the views of information at different levels differ in terms of both amount of detail and scale of activity, ranging from detailed information about small-scale activity (infrastructure level) to abstract information about large-scale activity (process level). The different views of information are as follows:

1. In the *infrastructure view*, information is seen from the point of view of the infrastructure underlying the collaboration system, consisting of files that contain records of objects and actions.

2. In the *system view*, information is seen from the point of view of the collaboration system, consisting of its information repositories, such as database tables and log files that contain records of objects and actions.

3. In the *user view*, information is seen from the point of view of the individual user, consisting of the objects of the collaboration system which the user interacts with using the actions provided by the collaboration system.

4. In the *collaboration view*, information is seen from the point of view of multiple users in collaboration with each other, consisting of the objects of the collaboration system which these users interact with using the actions provided by the collaboration system.

5. In the *task view*, information is seen from the point of view of multiple users performing tasks, consisting of multiple collaboration-level actions and objects belonging to these tasks.

6. In the *process view*, information is seen from the point of view of multiple users performing processes, consisting of multiple related tasks belonging to these processes.

Given these views of information at different levels of abstraction, all of which are based on the same underlying objects and actions, it is proposed that a source body of data can be transformed into high-level patterns of virtual collaboration through a *series of increasingly abstract intermediate-level patterns*. This abstraction implies a generalization from the specific detail of one view to less specific detail of another view. Based on the views of information presented in Figure 3.3 above, here a model of information consisting of six levels of patterns is proposed: the *Information Pyramid of Virtual Collaboration*, shown in Figure 3.4.

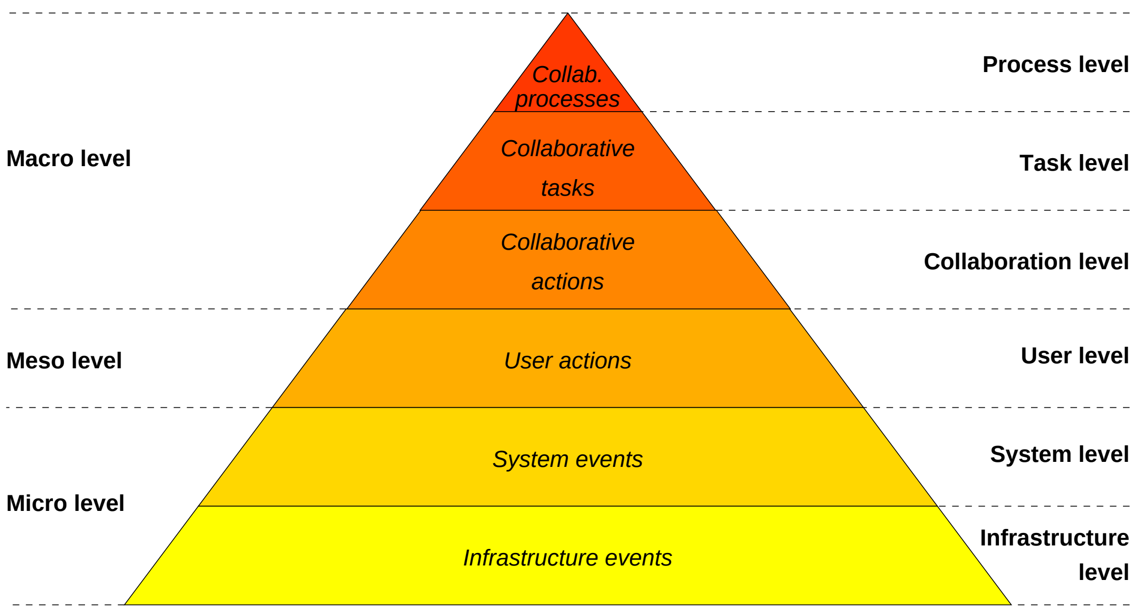


Figure 3.4: Information Pyramid of Virtual Collaboration with different levels of information

As with the views of information, at the bottom of the Information Pyramid is the most small-scale, detailed information, while at the top is the most large-scale, abstract information. This is expressed in the shape and colour of the figure. The shape of the figure suggests that the amount of information at higher levels is smaller, as it constitutes a higher level of abstraction. The different colours suggest that information at higher levels is *denser* than that at lower levels, in the sense that each unit of higher-level information corresponds to several units of lower-level information. From bottom up, the different levels contain following information:

1. **Infrastructure level:** This is the level of the underlying software infrastructure running "below" the collaboration system itself. In the case of a web-based collaboration system, for instance, the underlying infrastructure is a web server. At this level, objects are recorded in the files under the control of the underlying system. Actions are typically recorded as events occurring in the software infrastructure, such as web server access requests recorded in a web server log. Another example are records in the transaction log maintained by a database management system, in the case where a collaboration system operates on top of such a system.

   Action patterns at this level consist of actions and action context that correspond to events in the software infrastructure.

2. **System level:** This is the level of the collaboration system itself, through which

collaboration is carried out. Records of objects at this level are contained in the application data of the collaboration system, typically residing in files or database tables. Actions are the commands issued to the collaboration system.

Collaboration systems are typically structured as client-server systems, where multiple clients are served by one server. In this case, clients send service requests to a server, which then performs the requested actions. Records of such service requests, such as in a server log, constitute records of actions at this level. This information is of a larger scale than the corresponding information on the infrastructure level, so a single object or action on the system level usually corresponds to multiple objects or actions on the infrastructure level.

Action patterns at this level consist of action and action context that correspond to operations performed by the collaboration system.

3. **User level:** This is the level on which individual users operate. These users perform actions on objects residing in collaboration spaces. Objects at this level are the collaboration spaces and other objects contained in them, while actions at this level are the operations performed by users, such as for instance opening a document for reading. Objects at this level are often abstractions of corresponding objects at the system level. Likewise, actions at this level often correspond to multiple actions on the system level; i.e. a single action performed by the user may require the collaboration system to perform several system-level actions.

   Action patterns at this level consist of action and action context that correspond to operations performed by a single user.

4. **Collaboration level:** At this level, multiple users work in collaboration with each other. Objects at this level, as on the user level, are the collaboration spaces and other objects contained in them, while actions at this level are the operations performed by multiple users. Objects at this level mostly correspond closely to those at the user level. However, actions at this level are abstractions of multiple user-level actions.

   Action patterns at this level consist of action and action context that correspond to operations performed by groups of users.

5. **Task level:** At this level, larger-scale activity involving several lower-level actions takes place. Objects at this level are groupings of multiple lower-level objects, while actions at this level are the tasks performed by multiple users. These tasks consist of certain combinations of actions and objects from lower levels.

Action patterns at this level consist of action and action context that correspond to tasks performed by groups of users.

6. **Process level:** At this, the highest level of the Information Pyramid, collections of tasks are performed by groups of users. These constitute work processes, i.e. collections of related tasks. Objects at this level are combinations of multiple lower-level objects involved in the process. Actions at this level are collections of task-level actions.

   Action patterns at this level consist of action and action context that correspond to processes performed by groups of users.

A broad categorization of levels in the Information Pyramid is shown by the labels on the left hand side of Figure 3.4: micro level, meso level, and macro level. This categorization is centred on the user level, for it is here that the actual actions performed by users of the collaboration system take place. This level is designated as the *meso level* in this categorization. At levels below the meso level, multiple smaller-scale operations corresponding to each user action occur, thus the designation *micro level*. On the other hand, at levels above the meso level are aggregations of individual user actions into multi-user actions, tasks, and processes, thus the designation *macro level*.

Multi-layer models of data and/or information can be used to bridge disparities between desired and required levels of detail. A prominent example of such a multi-layer model is the ISO OSI Reference Model for networking (Tanenbaum, 1988). In this model, the desired level of detail is that of the communication between processes on the model's top layer, the application level. However, in order to bring this communication about, the required level of detail involves individual bits moving across a network medium such as a wire, which takes place on the model's bottom layer, the physical layer. By modeling this communication in seven stacked layers, the communication is conceptually simplified. The same principle applies to the Information Pyramid. It too consists of multiple layers with varying degree of detail. The desired level of detail is that of collaborative tasks and processes, at the top levels of the model, while the required level of detail is that of records of constituents of these tasks and processes, residing at the bottom levels of the model. As in the case of the OSI model, the modeling in multiple layers conceptually simplifies the mapping between the bottom and top layers.

Related work in the domain of workflow has been presented in (Weigand et al., 2000), which applies linguistic theory, in particular speech act theory (Searle, 1969) and Habermas' theory of communicative action (Habermas, 1981), to the analysis of communication and conversation. Analysis is performed on *workflow patterns* on different levels of abstraction.

The levels of the Information Pyramid are illustrated later on in this chapter with an example of information at each level. First, however, a graphical representation of action patterns is introduced.

3.3 Graphical Representation of Action Patterns

To fully describe an action pattern requires the specification of its action and action context, details of which are given in the following chapter. However, a simple graphical notation can be useful in conveying main features of an action pattern. Such a notation is presented here. The notation aims to both aid communication about the contents of an action pattern, and to focus on only its main features.

The definition of collaboration spaces, given in Chapter 2 on page 32, identifies several important concepts, namely: collaboration spaces, people, artefacts, communication channels, and joint activity. These five concepts constitute the main features that need to be represented in a graphical notation of action patterns.

In the review of representations of virtual collaboration processes in Chapter 2, the notations of the Collaborative Business Process Model were introduced. These include *MOO diagrams* for representing tasks, and *rich pictures* for representing processes (Hawryszkiewycz, 2000). These notations have representations for roles, actions, interactions, and artefacts, thus being close to the requirements for the notation needed here. The representation of action patterns can thus largely utilize these notations. Only the MOO diagramming notation needs to be slightly extended.

Here, the **extended MOO diagram**, or **EMOO diagram** for short, is introduced, which forms an extension of the original MOO diagramming notation of (Hawryszkiewycz, 2000). The following are the extensions to the original MOO diagramming notation added by the EMOO diagramming notation:

1. Defined defaults for actions.

2. Representation of collaboration spaces.

3. Refined representation for roles to distinguish between those occupied by a single person vs. those occupied by multiple people (designated here as *singleton roles* and *multi-roles*, respectively).

4. Refined representation for artefacts (documents) to distinguish between those that consist of a single item vs. those that consist of multiple ones (designated here as *singleton artefacts* and *multi-artefacts*, respectively).

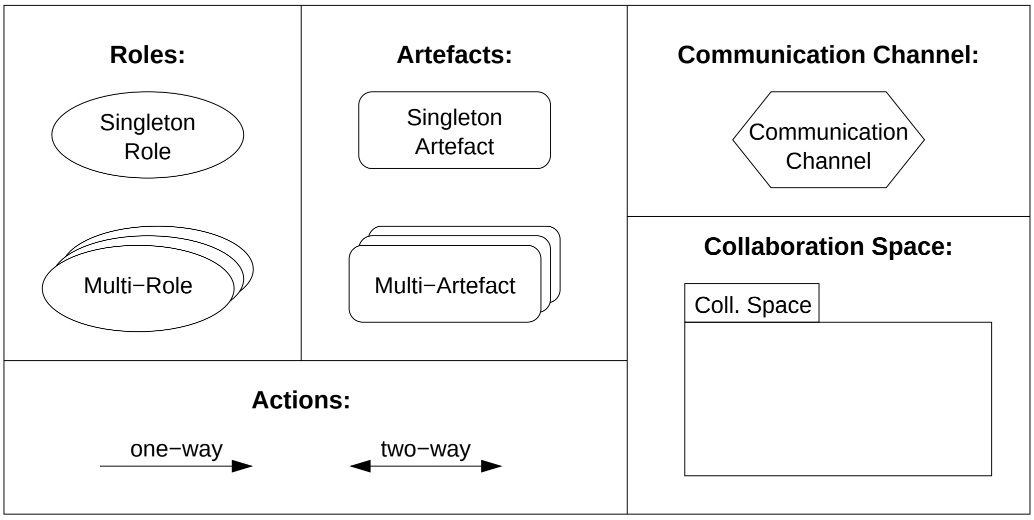


Figure 3.5: Modeling elements of EMOO diagrams

The distinction between singleton roles and multi-roles facilitates the description of situations of virtual collaboration in which it is important to know whether a single person or multiple people are involved, while the distinction between singleton artefacts and multi-artefacts facilitates representation of situations of virtual collaboration involving a large number of related artefacts which are to be treated as a collective entity. Graphical symbols for these various modeling elements are shown in Figure 3.5. Descriptions of the different modeling elements are listed below:

Role: An organizational role occupied by one or more people. A role occupied by only one user of a collaboration system is termed a *singleton role* and is represented by a single oval, labeled with the name of the role. A role occupied by multiple users of a collaboration system is termed a *multi-role* and is represented by three overlapping ovals, the one in front being labeled with the name of the role.

Artefact: A passive object or collection of objects containing information. An artefact consisting of only one object is termed a *singleton artefact* and is represented by a single rectangle with rounded corners, labeled with the name of the artefact. An artefact consisting of multiple objects is termed a *multi-artefact* and is represented by three overlapping rectangles with rounded corners, the one in front being labeled with the name of the artefact. Examples of artefacts include text documents, drawings, audio/video recordings, etc.

Communication channel: A facility for the exchange of messages, available to users of a collaboration space. A communication channel is represented by a hexagon, la-

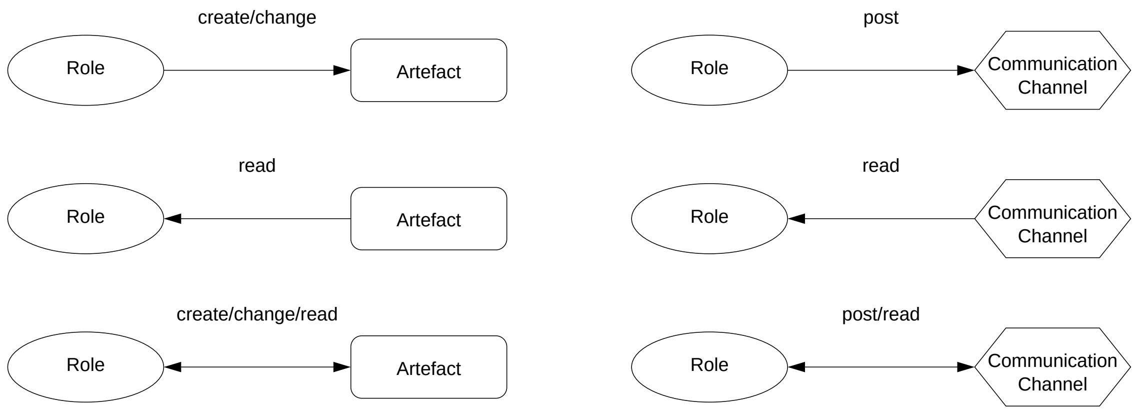


Figure 3.6: Default meanings of actions in EMOO diagrams

beled with the name of the communication channel. Examples of communication channels include discussion forums, text-based chat, audio or audio-video channels, etc.

Action: A function or operation that can be performed in a collaboration system. An action is represented by a single- or double-headed arrow connecting a role (the subject of the action) with an artefact or communication channel (the referent of the action), and may optionally be labeled with the name of the action. If the arrow is not labeled, a default meaning of the action is assumed (see below).

Collaboration space: A virtual space in which roles, artefacts, and communication channels may be placed, and in which actions may be performed. A collaboration space is represented by a rectangle, and is labeled with its name shown in a small rectangle placed flush left on top of the rectangle representing the collaboration space.

As mentioned above, the arrow representing an action may not be labeled. In this case, the arrow connecting subject and referent of the action it represents is assumed to take on a default meaning. Figure 3.6 shows the default meanings of different actions involving the modeling elements *role*, *artefact*, and *communication channel*. The figure only shows singleton roles and singleton artefacts, however the same meaning applies to actions involving multi-roles and multi-artefacts, respectively.

A simple example of an EMOO diagram corresponding to the MOO diagram of Figure 3.2 is shown in Figure 3.7. It shows one collaboration space (“Prepare-Report”), containing two roles (“Coordinator”, a singleton role, and “Writer”, a multi-role), two artefacts (“Final-Report”, a singleton artefact, and “Report-Parts”, a multi-artefact), one communication channel (“Discuss Report”), and five actions (the arrows connecting the roles with the artefacts and the communication channel, all assuming default meanings).

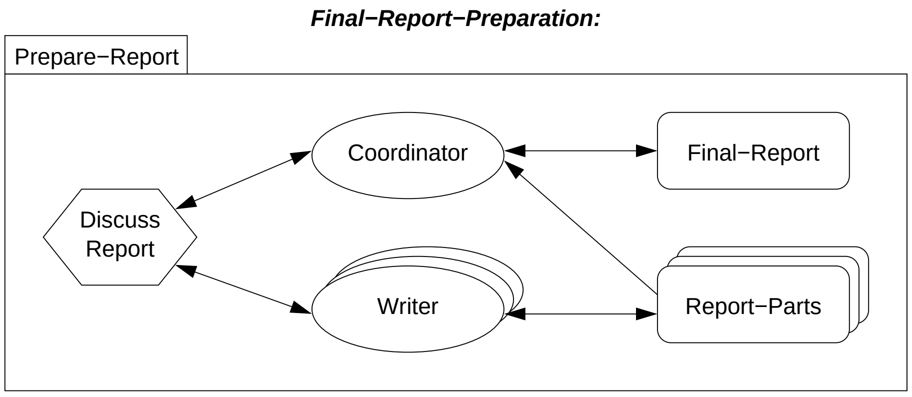


Figure 3.7: EMOO diagram of a report preparation action pattern

3.4 Example of Levels of Information

To illustrate the levels of the Information Pyramid, an example of patterns of virtual collaboration on different levels is presented below. The example is based on data obtained from the LIVENET collaboration system. As the main source of information in LIVENET is on the system level, the example illustrates patterns of virtual collaboration starting from that level up to the process level. The example is related to a specific virtual collaboration process, concerned with product concept development, and shows patterns of virtual collaboration of parts of that same process at different levels of the Information Pyramid up to the process level where the process as a whole is shown. The example is intended to illustrate the different information at different levels of the Information Pyramid, and their correspondence across levels. It is specifically not concerned with explaining how higher-level information is derived from lower-level information, as this is the subject of the following chapter.

3.4.1 System Level

In the LIVENET system, actions performed by the LIVENET server are recorded in a server log. Each log entry records the action performed by the LIVENET client, context information including subject, referent, location, and time, and any other attributes of the action that may be supplied. An example taken from the LIVENET log for three consecutive system-level actions is shown below[3]:

[3]Identifying information, such as actual user and group names, have been changed in this and all following examples to preserve the anonymity of the users involved.

Field	Attribute
1	*Log-Id*
2	*Timestamp*
3	*Session-Id*
4	*Workgroup*
5	*Workspace*
6	*Workspace-Owner*
7	*User*
8	*Role*
9	*Action*
10 . . . 19	*Action Attributes*

Table 3.1: Fields in LIVENET's system-level log records

```
[84989|2000.08.26 19:12:05.103|4325|Group3|Prepare-Report|John.Smith|
  Mary.Lamb|Writer|get_block_tree|Prepare-Report_John.Smith|
  Discuss-Report]

[84990|2000.08.26 19:12:05.514|4325|Group3|Prepare-Report|John.Smith|
  Mary.Lamb|Writer|add_statement|1094|0|Reminder|Please upload your
  part of the impact and activity change table before 9 PM tomorrow.
  Please also download and review all others part before the meeting
  on the coming Monday. We will have to complete the Milestone by
  next Monday.|null]

[84991|2000.08.26 19:12:06.515|4325|Group3|Prepare-Report|John.Smith|
  Mary.Lamb|Writer|get_block_tree|Prepare-Report_John.Smith|
  Discuss-Report]
```

Each of these records consists of the fields shown in Table 3.1, where adjacent fields are separated by a vertical bar ('|'). Up to ten action attributes may be present, but the actual number of attributes depends on the action.

The three records shown correspond to three actions performed by the LIVENET server over a time period of about 1.5 seconds on 26/8/2000 (the actions with the log-id numbers 84989–84991). The actions were *get_block_tree* (an action for retrieving a list of statements posted in a discussion forum), followed by *add_statement* (an action for posting a statement in a discussion forum), followed by another *get_block_tree* action.

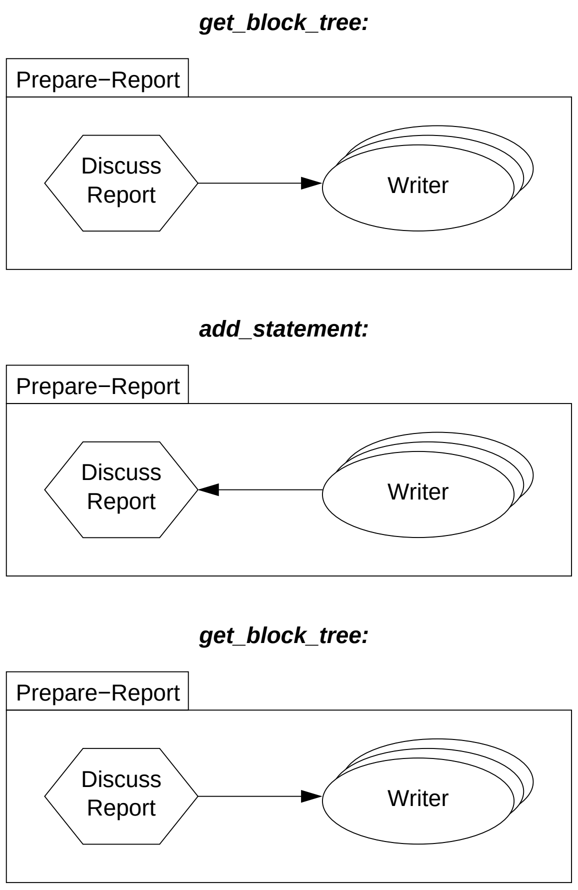


Figure 3.8: EMOO diagrams of three consecutive system-level action patterns *get_block_tree*, *add_statement* and *get_block_tree*

They were issued in session 4325 from within workgroup *Group3*, in workspace *Prepare-Report* owned by *John.Smith*. The actions were performed by *Mary.Lamb*, taking the role of *Writer* in the given workspace. For each of these actions, a system-level action pattern can be derived, including both information from the log record and from the LIVENET application database (for information on actions and objects, respectively). EMOO diagrams of these three system-level action patterns are shown in Figure 3.8.

The action patterns obtained from records of the collaboration system's actions are very detailed, but only represent very small-scale activity. On the next level of the Information Pyramid, more abstract action patterns are derived therefrom.

3.4.2 User Level

A user-level action in LIVENET is an action performed by a single user. Examples of this include: opening a document, entering a workspace, posting a discussion statement,

Attribute	**Value**
Session-Id	4325
Action-Number	4
Timestamp	2000.08.26 19:12:05
Begin-Action-Id	84989
End-Action-Id	84991
Workgroup	Group3
Workspace	Prepare-Report
Workspace-Owner	John.Smith
User	Mary.Lamb
Role	Writer
Action	Post-Discussion-Statement
Disc-Forum-Id	1094
Parent-Stmt-No	0

Table 3.2: Attributes of an instance of user-level action pattern *Post-Discussion-Statement*

etc. User-level actions and action patterns are not logged in LIVENET, thus they can only be *derived* from corresponding system-level actions and action patterns. Details of this derivation process are presented in the following chapter; here an example of a derived user-level action pattern is presented.

From the three system-level action patterns shown above, a single user-level action pattern can be derived, as shown in Table 3.2. This action pattern shows that in session 4325, action number 4 took place at about 7:12PM on 26/8/2000. The user-level action corresponds to system-level actions 84989 to 84991 (these are the three system-level actions represented in the three system-level action patterns shown above). It took place within workgroup *Group3*, in workspace *Prepare-Report* owned by *John.Smith*. The user-level action was performed by user *Mary.Lamb*, taking the role of *Writer* in the given workspace, and the action was *Post-Discussion-Statement*. A number of action attributes (*Disc-Forum-Id* and *Parent-Stmt-No*) are also given.

The user action *Post-Discussion-Statement* in this case corresponds to three system-level action patterns, because the collaboration system, LIVENET, carries out three actions each time a statement is posted in a discussion forum: first it obtains a list of statements in the discussion forum, then it adds the new statement, then it obtains an updated list of discussion statements. The rationale for this particular sequence of actions is unknown, and is not relevant. What matters is that anytime this sequence of actions takes

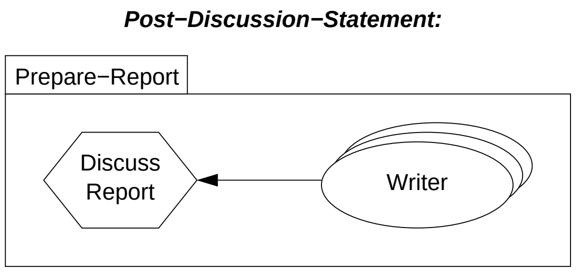


Figure 3.9: EMOO diagram of user-level action pattern *Post-Discussion-Statement*

place in the same session, a user-level action pattern can be derived which corresponds to the user-level action of posting a discussion statement. That is, the given sequence of system-level action patterns can be *transformed* to a single user-level action pattern.

The EMOO diagram in Figure 3.9 represents the *Post-Discussion-Statement* user-level action pattern presented in this example. It shows that the role *Writer* (a multi-role) is connected with the discussion forum *Discuss Report* through a posting action (shown by the arrow pointing from the role to the discussion forum).

This action pattern is that of a single user at a single point in time. The next step of abstraction is to consider collections of such action patterns.

3.4.3 Collaboration Level

A collaboration-level action in LIVENET is an action performed by a group of users. A collaboration-level action pattern corresponds to a collection of user-level action patterns with a (partially) shared action context. The part of the action context that is shared may be time (actions taking place in temporal proximity), location (actions taking place in spacial proximity, where the space is understood to be virtual), or some object being affected by the action pattern (a document jointly worked on; a discussion forum where a joint discussion takes place; etc.).

To continue the earlier example, a group of user-level action patterns within the shared context of a given discussion forum (the action's referent) together constitute the collaboration-level action pattern *Group-Discussion*. It corresponds to a number of user-level action patterns related to the given discussion forum. In the case of LIVENET, the user-level action patterns which contribute to the *Group-Discussion* action pattern include *Post-Discussion-Statement* (posting a statement) and *Open-Discussion-Statement* (reading a statement). Attributes of the corresponding collaboration-level action pattern are shown in Table 3.3.

This action pattern shows that the collaboration-level action with id 736 took place

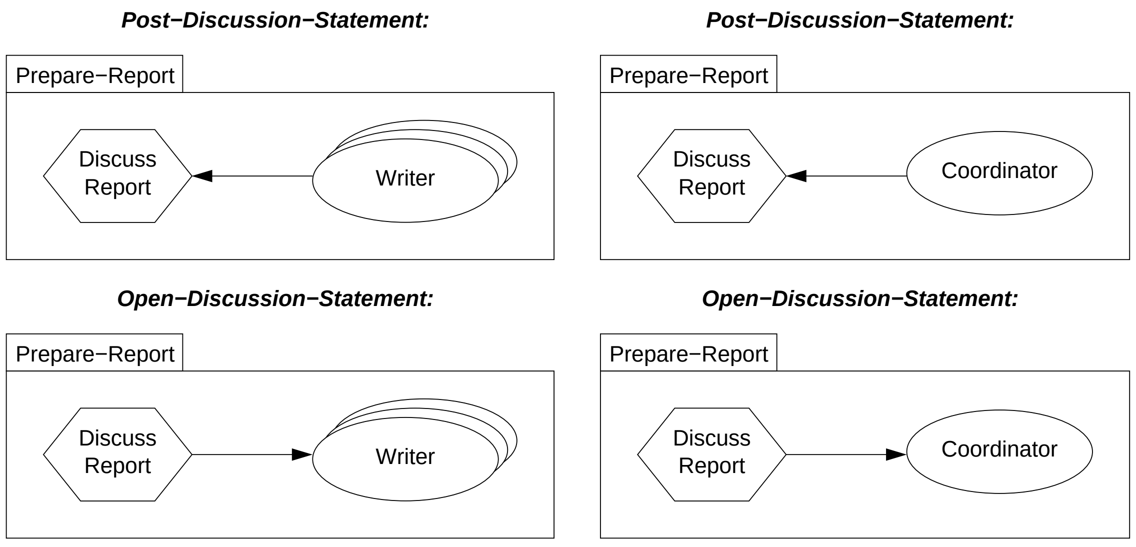
Attribute	**Value**
Action-Id	736
Begin-Timestamp	2000.08.21 22:08:17
End-Timestamp	2000.11.01 16:58:23
Workgroup	Group3
Workspace	Prepare-Report
Workspace-Owner	John.Smith
Users	John.Smith, Mary.Lamb, Paul.Jones, Helen.Blake
Action	Group-Discussion
Num-Posts	34
Num-Reads	84
Read-Post-Ratio	2.47
Posts-Per-Day	0.48
Reads-Per-Day	1.18
Avg-Thread-Size	1.82
Avg-Thread-Depth	1.73

Table 3.3: Attributes of an instance of collaboration-level action pattern *Group-Discussion*

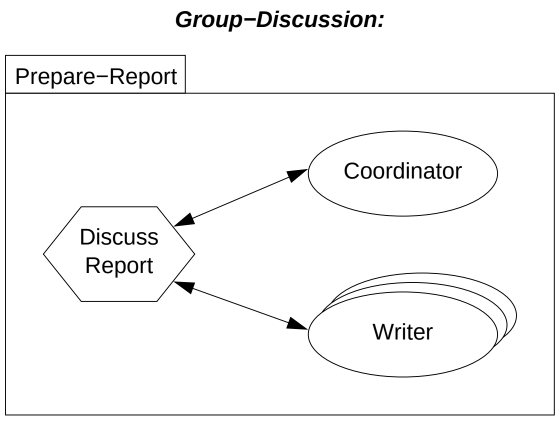
between the dates 21/8/2000 and 1/11/2000 within workgroup *Group3*, in workspace *Prepare-Report* owned by *John.Smith*. The action was of type *Group-Discussion*, involving the four users *John.Smith*, *Mary.Lamb*, *Paul.Jones*, and *Helen.Blake*. Several action attributes are included, in this case *discussion metrics*: the total number of *Post-Discussion-Statement* actions (34); the total number of *Open-Discussion-Statement* actions (84); the read/post ratio (2.47); the average number of messages posted per day (0.48); the average number of messages read per day (1.18); the average size of discussion threads, in number of statements (1.82); and the average discussion thread depth (1.73), being the number of levels of replies in each thread.

Besides attributes about action context, such as time, location, etc., the inclusion of discussion metrics in this record expresses something about the nature of the discussion taking place among this particular group of users, such as the intensity of the discussion (being a measure of the numbers of posts and reads per day); the depth of the discussion tree (being a measure of thread depth); etc. These discussion metrics can provide additional information to help categorize different instances of the *Group-Discussion* action pattern.

Figure 3.10 (b) shows an EMOO diagram of the *Group-Discussion* collaboration-



(a) User-level action patterns



(b) Collaboration-level action pattern

Figure 3.10: EMOO diagram of collaboration-level action pattern *Group-Discussion* as an aggregation of *Open-Discussion-Statement* and *Post-Discussion-Statement* user-level action patterns involving roles *Writer* and *Coordinator*

level action pattern presented in this example. It shows that two roles are involved in the discussion (of which *Writer* is a multi-role). Both of the roles have read and post access to the discussion forum, which thus forms an aggregation of several individual *Open-Discussion-Statement* and *Post-Discussion-Statement* user-level action patterns. Part (a) of the figure shows the four individual user-level action patterns that are aggregated into the collaboration-level action pattern.

3.4.4 Task Level

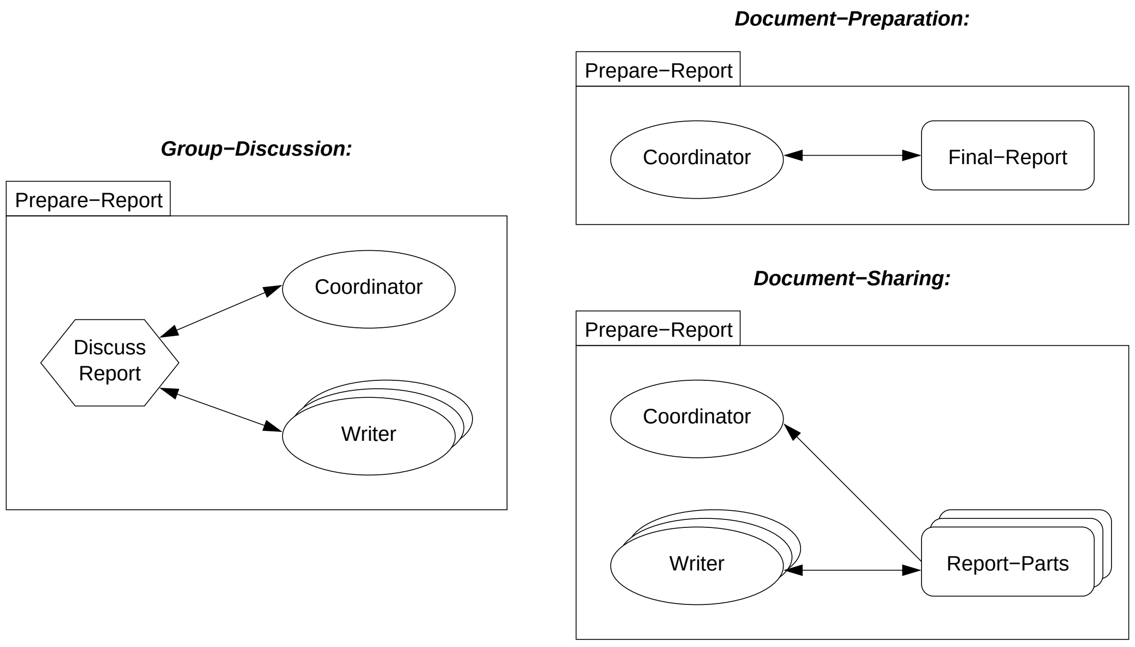
A task-level action in LIVENET is a larger-scale activity (compared to a collaboration-level action) and is performed by a group of users. A task-level action pattern corresponds to the combination of two or more collaboration-level action patterns.

An example of a task is that of joint report preparation. This is an activity which involves several collaboration-level actions: it may start out with a discussion of the format and structure of the report, followed by individual document preparation work. This may then lead to document sharing and review, before integrating the separate report pieces into the whole report document.

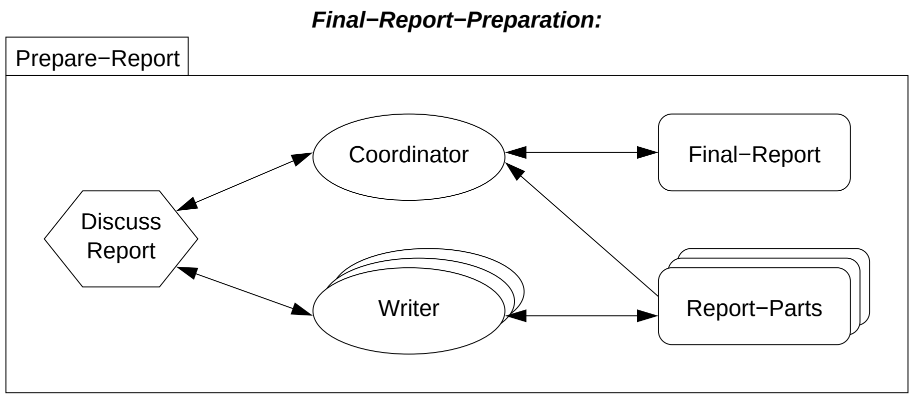
In terms of the action patterns involved, this task may consist of a combination of *Group-Discussion*, *Document-Sharing*, and *Document-Preparation* collaboration-level action patterns. They are combined through the subjects (i.e. roles) involved in those action patterns, linking the collaboration-level action patterns together into the *Final-Report-Preparation* task-level action pattern. An EMOO diagram of this action pattern, combining the earlier *Group-Discussion* action pattern from Figure 3.10 with a *Document-Sharing* and a *Document-Preparation* action pattern, is shown in Figure 3.11. Part (a) of the figure shows the constituent action patterns, while the task-level action pattern itself is shown in part (b) of the figure. Here the *Document-Sharing* collaboration-level action pattern involves both the *Coordinator* and *Writer* roles, and is mediated through the *Report-Part* artefact (a multi-artefact, with one artefact for each report part). Artefact access within the *Document-Sharing* collaboration-level action pattern differs between the two roles: while the *Writer* role has both read and write access, the *Coordinator* role has only read access. Lastly, the *Final-Report* artefact constitutes the task's final outcome, and is produced by the *Coordinator* role through the *Document-Preparation* action pattern.

3.4.5 Process Level

A process-level action is the largest-scale activity in the Information Pyramid, and is performed by a group of users. A process-level action pattern consists of a number of



(a) Collaboration-level action patterns



(b) Task-level action pattern

Figure 3.11: EMOO diagram of task-level action pattern *Final-Report-Preparation* as a combination of action patterns *Group-Discussion*, *Document-Sharing*, and *Document-Preparation*.

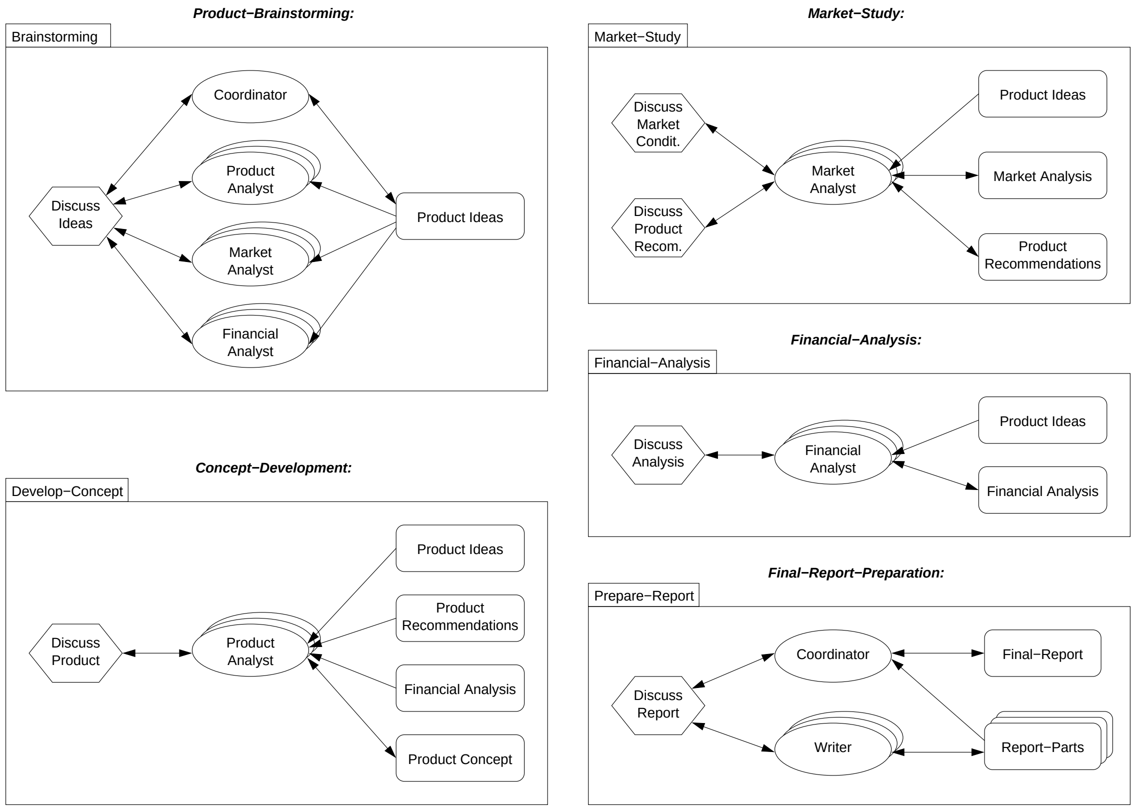
task-level action patterns.

Following on from the example of joint report preparation, this task may be part of a process concerned with developing concepts for new products. The whole process may consist of several tasks, including: brainstorming ideas for new products, market study, financial analysis, development of a selected product concept, and finally preparation of a report with the results of the individual tasks.

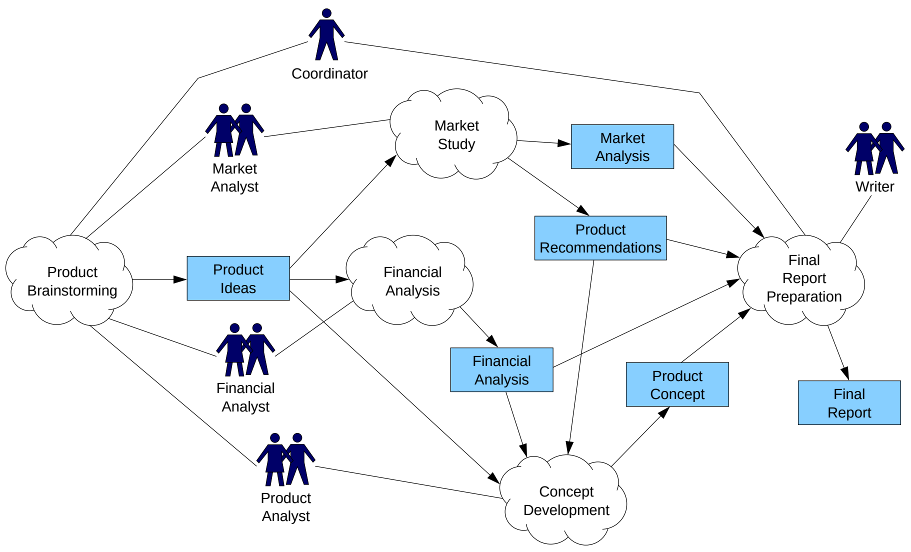
In terms of the action patterns involved, this process combines the five task-level action patterns *Product-Brainstorming*, *Market-Study*, *Financial-Analysis*, *Concept-Development*, and *Final-Report-Preparation* which was already shown above. Each of these task-level action patterns takes place in its own collaboration space and involves a number of roles, communication channels and artefacts. All roles are involved in more than one task-level action pattern, as are most of the artefacts. EMOO diagrams of these five task-level action patterns are shown in Figure 3.12 (a). Together these task-level action patterns constitute the process-level action pattern *Product-Concept-Development*, shown in the form of a rich picture in Figure 3.12 (b).

———— . ————

The above example has illustrated the different levels of the Information Pyramid of Virtual Collaboration, from the system level up to the process level, showing instances of action patterns at each of these levels. The example showed that action patterns on a given level (with the exception of the lowest level) are aggregations of action patterns on the level below. Thus an instance of a higher-level action pattern corresponds to multiple instances of lower-level action patterns. In this way there is a *chain of correspondences* of action patterns from the lowest level to the highest level of the Information Pyramid. This bridges the disparity mentioned at the beginning of this chapter, in terms of amount of detail and scale of the activity between the source data and the patterns of virtual collaboration sought. In the case of the above example, the chain of corresponding action patterns across levels is shown in Figure 3.13. It shows the correspondence of two action patterns on the system level (add_statement and get_block_tree) to the user-level action pattern Post-Discussion-Statement (the correspondence being represented by the line connecting the names of two action patterns). In turn, there is a correspondence of two user-level action patterns (Post-Discussion-Statement and Open-Discussion-Statement) to the collaboration-level action pattern Group-Discussion. This chain of correspondences continues step-by-step until it reaches the process level, and the action pattern Product-Concept-Development.



(a) Task-level action patterns



(b) Process-level action pattern *Product-Concept-Development*

Figure 3.12: EMOO diagrams of five task-level action patterns and rich picture of the corresponding process-level action pattern *Product-Concept-Development*

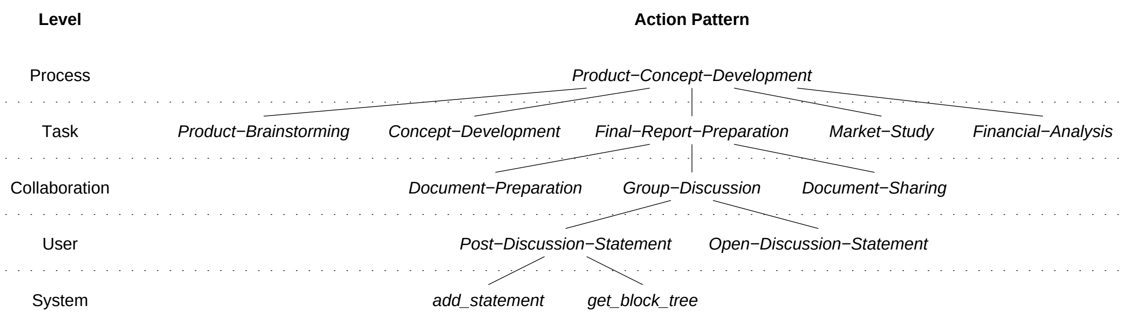


Figure 3.13: Chain of correspondences of action patterns from system level to process level of the Information Pyramid

3.5 Summary

This chapter has discussed the modeling of patterns of virtual collaboration. It started by considering what a pattern is, providing a definition of the term.

Next, the Information Pyramid of Virtual Collaboration was proposed as a model of information related to collaboration systems and the virtual collaboration carried out through them. This information model suggests that information about virtual collaboration can be considered at six different levels of abstraction, presenting a different *view* of information at each level, ranging from small-scale activities to entire tasks and processes. In this model, two main kinds of information exist, *objects* and *actions*. An action extended by an *action context* is an *action pattern*, which describes an activity of virtual collaboration.

The graphical notation of EMOO (extended MOO) diagrams was proposed to represent action patterns. This was followed by an illustrative example of action patterns at the different levels of the Information Pyramid of Virtual Collaboration for an actual collaboration system.

The next chapter carries on from here to deal with the representation of information and the pattern derivation process in more detail.

Chapter 4

Deriving Patterns of Virtual Collaboration

The previous chapter has introduced the notion of *patterns of virtual collaboration*, and proposed a multi-level model of information, the Information Pyramid of Virtual Collaboration (cf. Figure 3.4 on page 72). The present chapter carries on from there to consider how patterns of virtual collaboration can be obtained from information on lower levels of the Information Pyramid and be transformed onto progressively higher levels.

This thesis argues that the successful derivation of patterns of virtual collaboration relies on the combination of both of these: an information model for representing patterns of virtual collaboration, namely the Information Pyramid presented in the previous chapter; and a process for modeling and deriving these patterns, namely the methods and framework presented in the current chapter.

4.1 Information Derivation

The extraction of patterns of virtual collaboration is crucially dependent on the availability of information related to this collaboration, namely about objects, actions, and action context (cf. the discussion in Section 3.2). As the source of this information is data collected by the collaboration systems themselves, these systems need to collect sufficient data to enable the extraction of patterns. The issue of designing data collection for collaboration systems is explored in more detail in Section 4.5. For now, however, it shall be acknowledged that different existing systems may collect source data on different levels of the Information Pyramid.

Conceptually, the Information Pyramid consists of six levels. However, not every collaboration system will collect data on all of the levels, and most will usually only collect

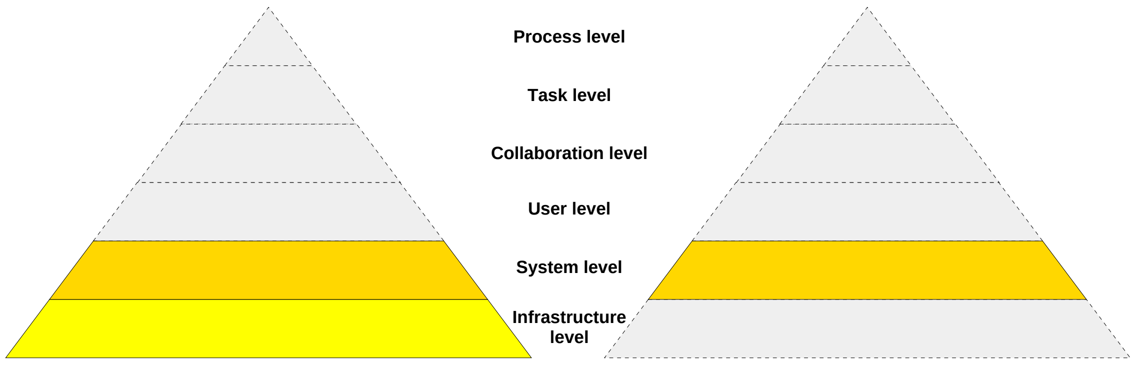


Figure 4.1: Information Pyramid for two actual collaboration systems, LIVENET (left) and TEAMROOMS (right)

data on one or two levels. For instance, a traditional client-server collaboration system may only collect data on the system level, but not on the infrastructure level. On the other hand, a web-based collaboration system may collect data on the infrastructure level and the system level. In yet another case, the collection of higher-level data may have been considered during the design of a collaboration system, and it may collect data at the user level. In most cases, however, the data source is at the micro level of the Information Pyramid, while meso and macro level data is usually absent. As an example, consider the situation of the LIVENET and TEAMROOMS systems, illustrated in Figure 4.1. LIVENET is a web-based system and thus the web server records infrastructure-level events, while the LIVENET server records system-level events. In addition, the system level holds the application data in the form of a workspace database. The meso and macro levels of the Information Pyramid, however, are absent. The TEAMROOMS system, on the other hand, is not web-based but is a conventional client-server system. Its only data source thus is at the system level, consisting of application data, while the infrastructure level as well as meso and macro levels are absent.

However, as mentioned before in Chapter 3, information on upper levels can be *derived* from information on lower levels, producing an Information Pyramid that is complete from the level of the data source up to the top level.

Here it is proposed that performing derivations of higher-level information involves two main steps:

1. Describing information at a given level.

2. Performing a mapping which takes information on one level and converts it to a form on the next-higher level.

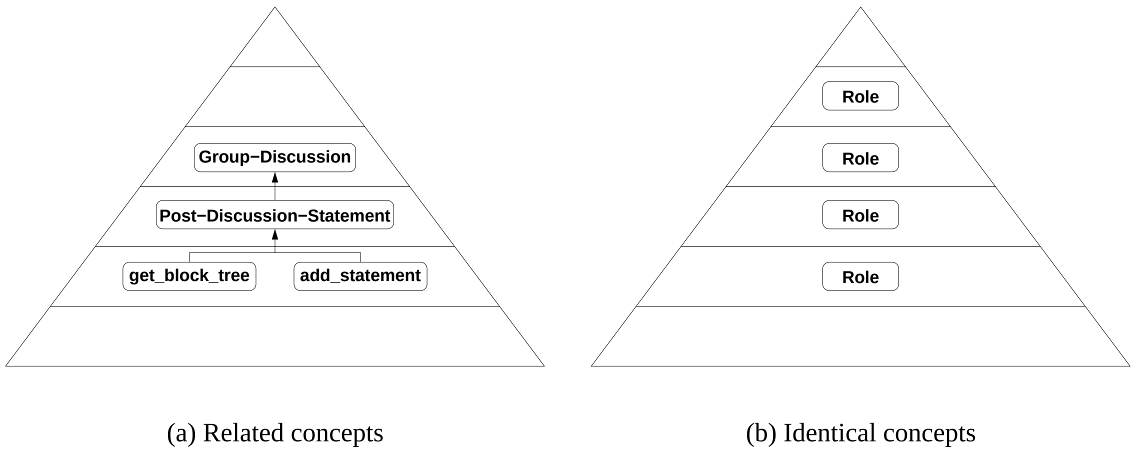


Figure 4.2: Related and identical concepts on different levels of the Information Pyramid

The first step entails *model building*, i.e. the specification of a representation of a set of entities, namely the information about virtual collaboration. For this, the definition of information in collaboration systems from Section 3.2 provides the basis. The second step entails *model transformation*, i.e. the specification of mappings between instances of models in different forms. Principles for mapping models are developed in this chapter.

4.1.1 Ontologies

In order to describe and map between different levels of the Information Pyramid of Virtual Collaboration, it is necessary to utilize a set of common, well-defined *concepts*. This set of concepts needs to express entities and actions associated with *all* levels of the modeled collaboration system, for following two reasons: firstly, as many concepts on one level are related to concepts on a neighbouring level, it follows that all levels need to be included to allow such relationships of concepts to be expressed. For instance, it was seen earlier that collections of instances of one or more action patterns on a given level may map to an instance of an action pattern on the next-higher level. This is illustrated in Figure 4.2 (a), showing a number of concepts on three different levels of the Information Pyramid being related to one another (this corresponds to the example presented earlier in Figures 3.8, 3.9 and 3.10). Secondly, as certain concepts are used across two or more levels, it is necessary that a single representation of the concepts exists, so as to avoid problems of both synonyms and homonyms, and thereby to ensure semantic consistency. For instance, the concept of a role may be used on multiple levels, yet have the same meaning. This is illustrated in Figure 4.2 (b), where the concept Role is present on four levels of the Information Pyramid.

It is suggested here that these requirements can be suitably satisfied by defining a meta-model, or *ontology* of these concepts. Ontologies have been used in the field of artificial intelligence for over a decade, for knowledge sharing and reuse. In that context, the term ontology has been defined as “an explicit specification of a conceptualization”[1] (Gruber, 1993). Another definition is given in (Sowa, 2000):

> The subject of *ontology* is the study of the *categories* of things that exist or may exist in some domain. The product of such a study, called *an ontology*, is a catalog of the types of things that are assumed to exist in a domain of interest *D* from the perspective of a person who uses a language *L* for the purpose of talking about *D*.

More recently, ontologies have found application in various other areas of the computing sciences (Gruninger and Lee, 2002).

Every ontology describes a certain *domain*, the *universe of discourse*. Collaboration systems have over the past decade emerged as a separate class of CSCW systems. The concepts supported by this class of systems, and the information provided by them, constitute the universe of discourse under investigation here. By specifying this universe of discourse in the form of an ontology, it becomes possible to bridge a number of separate conceptual realms. It was seen earlier that the Information Pyramid of Virtual Collaboration spans six levels, each having its own view of the information comprised within it, i.e. its own conceptual realm (cf. Figure 3.3 on page 68). Each of these could conceivably be specified in its own terms; for instance, the system level could be specified as an entity-relationship diagram of the database structure, accompanied by an event model of the system events. Other levels could similarly have their own specification, using their own modeling notations. However, relating concepts across levels, which as mentioned above is an important requirement, would then not be possible without employing some kind of intermediary mapping level or other approach to bridge these separate representations. The approach of using an ontology, on the other hand, makes it possible to specify, and then relate concepts from, all levels using a common notation and terminology.

Within the computing sciences today, ontologies are being used in mainly three capacities: for communication, for computational inference, and for reuse and organization of knowledge (Gruninger and Lee, 2002). In the present context, ontologies are mainly used in the first and third capacity: to communicate meaning, and to reuse and organize knowledge (although, in line with the meanings of these terms defined earlier, here the term “information” is used, rather than “knowledge”).

[1]The term “ontology” was originally coined in philosophy, where it means “the theory or study of being as such; i.e., of the basic characteristics of all reality”. (Source: Encyclopædia Britannica)

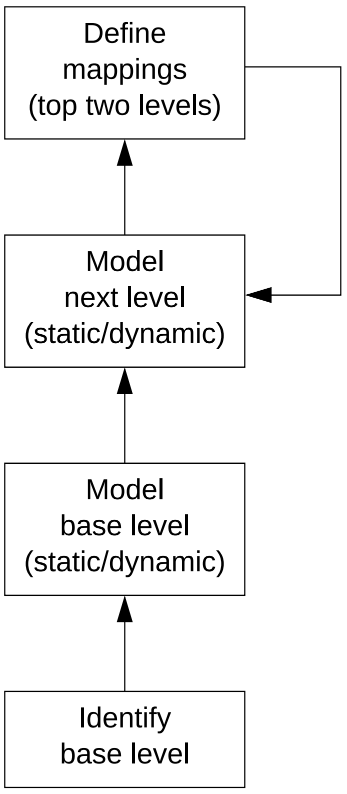

Define mappings (top two levels)
Model next level (static/dynamic)
Model base level (static/dynamic)
Identify base level

Figure 4.3: Modeling method for deriving information in the Information Pyramid

4.1.2 Modeling Method

For creating models and mappings between models for a given collaboration system, the following method is proposed, as illustrated in Figure 4.3:

Step 1: Identify the highest-level source of information available. This will constitute the *base level* of the Information Pyramid for the given collaboration system.

Step 2: Model the base level by modeling its static and dynamic concepts.

Step 2.1: Identify objects.

Step 2.2: Identify actions.

Step 2.3: Identify action patterns.

Step 2.4: Specify concepts.

Step 3: Model the next-higher level in the same manner; the modeling of concepts on higher levels may be based on corresponding concepts on lower levels.

Step 3.1: Identify unchanged/modified/new objects.

Step 3.2: Identify unchanged/modified/new actions.

Step 3.3: Identify unchanged/modified/new action patterns.

Step 3.4: Specify concepts.

Step 4: Define mappings between concepts on the two levels just modeled.

Step 4.1: Identify source and target concepts and attributes.

Step 4.2: Identify mapping constraints.

Step 4.3: Specify mappings.

Step 4.4: Define mapping functions.

Steps 5 and above: Repeat steps 3 and 4 until the top level is reached.

Each of the models and mappings adds to an evolving ontology, until the resulting ontology covers all levels of the Information Pyramid from the base level up, as well as the mappings between them. This is illustrated in Figure 4.4: while a different model and a different mapping is needed for different levels, a common ontology is used for representing all of the models and mappings.

4.1.3 Knowledge Model of the Ontology

Before entering into details on how the ontology for the models and mappings is specified, a few explanatory words about ontologies and their representation are in order.

The principal components of an ontology are *concepts* and their *relationships*. Concepts correspond to the entities that make up the universe of discourse, while relationships establish how these entities are related to one another. *Properties*, or *attributes*, capture detailed aspects of the concepts. Ontologies may also make use of various other constructs to specify certain aspects of the universe of discourse: *constraints*, *axioms*, *functions*, etc. (for further detail refer to (Gruber, 1993)).

To express an ontology, it is usual to employ a *knowledge model*. A knowledge model provides certain constructs to enable the specification of the universe of discourse that is the subject of the ontology. For the representation of the concepts related to the Information Pyramid of Virtual Collaboration, an existing knowledge model is employed. This is the knowledge model of Protégé-2000, an integrated knowledge-base development and management system (Noy et al., 2000); this knowledge model is similar to the widely-used OKBC knowledge model (Chaudhri et al., 1998). Protégé-2000 ontologies are specified in an extended form of the CLIPS language, a Lisp-like language which is

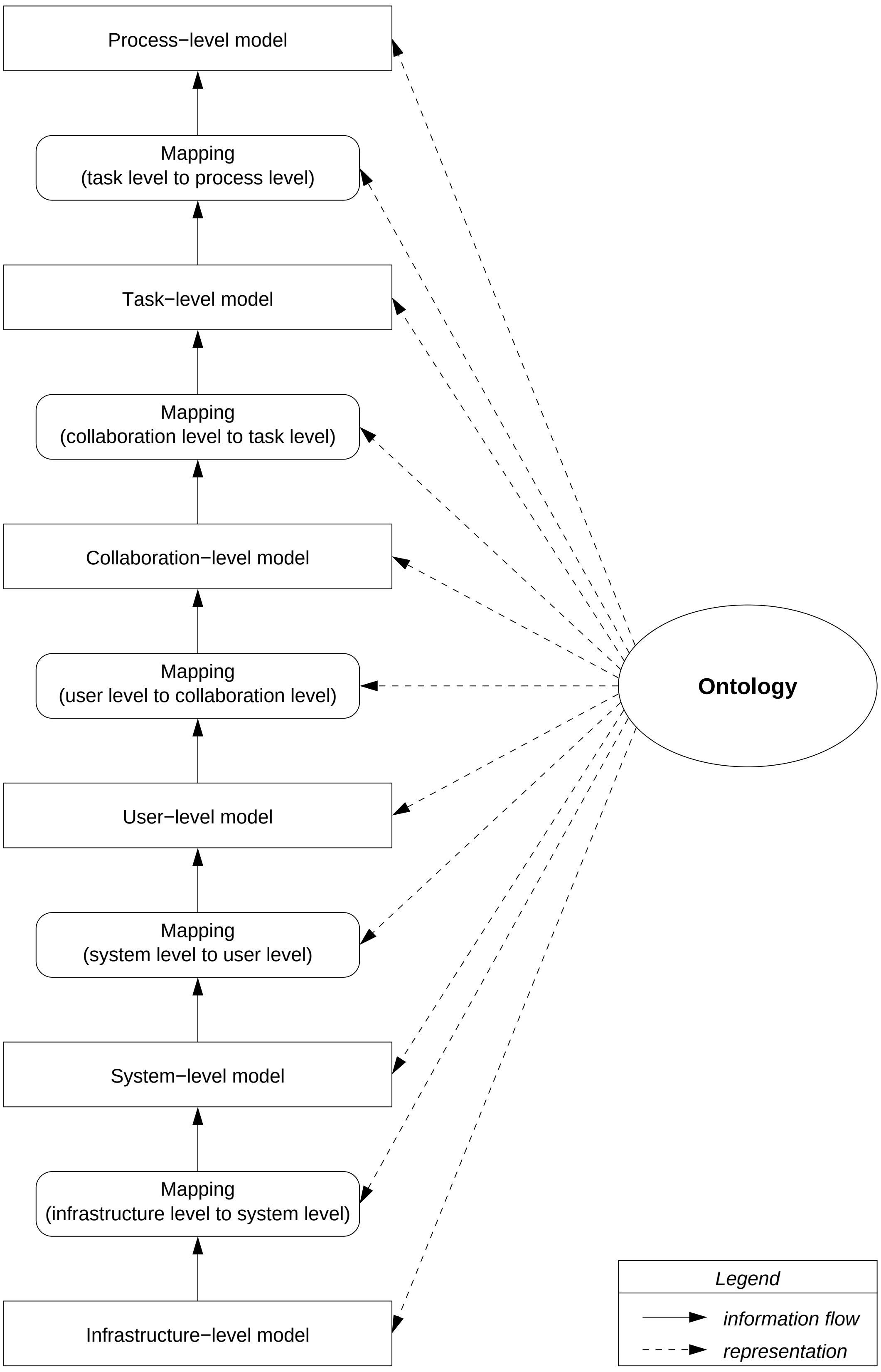

Process−level model
Mapping (task level to process level)
Task−level model
Mapping (collaboration level to task level)
Collaboration−level model
Mapping (user level to collaboration level)
User−level model
Mapping (system level to user level)
System−level model
Mapping (infrastructure level to system level)
Infrastructure−level model
Ontology
Legend
information flow
representation

Figure 4.4: Models and mappings for deriving information in the Information Pyramid

part of the CLIPS expert system shell (Giarratano and Riley, 1998). Protégé-2000 uses an object-oriented representation of concepts which employs following main modeling constructs:

Classes represent entities in the universe of discourse, i.e. concepts. For instance, the class `User` may be used to represent the concept of a collaboration system's user, i.e. a person involved in collaboration on the computer. Classes can have *specializations*, i.e. *subclasses* which inherit from their superclass, or in the case of multiple inheritance, from all their superclasses. Thus the set of classes in a given ontology forms a *taxonomic hierarchy*. A class whose instances are themselves classes is called a *metaclass*. Each class has a *role*, which indicates how the class may be used: *abstract* classes cannot be instantiated directly, while *concrete* classes may be instantiated.

Instances, then, are instantiations of a class. Classes and instances are thus similar to types and variables in programming languages: classes are like types, while instances are like variables of a given type. To follow the previous example, one instance of the `User` class could represent the user with the name `John Smith`. Instances inherit all attributes of the class they instantiate, as well as all of that class's ancestors (i.e. its superclass(es), recursively to the root of the class hierarchy).

Slots are attributes of classes and instances. They contain the details of the represented concepts. For example, the class `User` could have a slot `email-address` to hold the value(s) of the user's email address(es), while the instance of this class representing user `John Smith` could have, say, the two values `john.smith@xyz.com` and `john@yahoo.com` in this slot.

Facets are simple constraints imposed on slots. They can specify minimum and maximum cardinalities of the slot, as well as limit the range of allowable values. For instance, the slot `email-address` could have a facet that specifies the minimum cardinality to be 0 and the maximum to be unbounded, thus making it optional for a user to have an email address, but allowing multiple addresses.

Constraints, finally, are limitations or rules imposed on concepts and attributes in the ontology. As opposed to facets which are simple rules on the allowed values of a slot, constraints in the general sense allow the specification of complex conditions, such as involving multiple slots from different classes. For instance, a constraint may be specified that requires users who are owners of workspaces to have an email address that has the domain name `xyz.com`.

The above example of the concept `User` is illustrated in Figure 4.5, which also introduces the graphical notation used for representing these modeling concepts. The box with the rounded corners represents a class, while the box with sharp corners represents an instance. The two boxes are divided into two parts: for classes, the upper part identi-

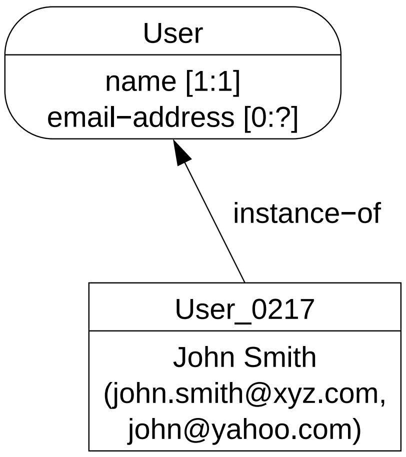


Figure 4.5: Part of an ontology, showing a class, instance, slots, and facets for representing users in a collaboration system

fies the class, while the lower part identifies any slots (and cardinality facets, if any); for instances, the upper part identifies the instance (by its class name and an instance number suffix), while the lower part shows values of slots. Here, the slot `email-address` of class `User` has the facet `[0:?]`, which specifies the slot's cardinality constraint (lower bound 0, no upper bound); while the shown instance of this class has two values for this slot.

4.1.4 Notation for Ontology Specification

The modeling concepts of ontologies just introduced are used in specifying concepts in the target domain. These are typically represented in textual notation, and may be supplemented with a graphical notation to facilitate understanding, such as the one just introduced. A number of textual notations are available; many of these employ a Lisp-like syntax, while more recently XML (the Extensible Markup Language) has been used for this purpose too. Here a slightly simplified form of the notation used by Protégé-2000 is adopted. This notation is Lisp-like, i.e. it uses nested lists with expressions, such as predicates or functions, in a prefix notation.

An example of the specification of class `User`, which was shown in graphical form in Figure 4.5, is given below (note that the line numbers shown to the left are not part of the class specification):

```
1 | (defclass User "A user of a collaboration system."
2 |    (is-a :THING)
3 |    (role concrete)
4 |    (single-slot name
5        (type STRING)
6        (cardinality 1 1))
7     (multislot email-address
8        (type STRING)
9        (cardinality 0 ?VARIABLE)))
```

Here, on line 1, `defclass` is a function that defines a new class, `User` is the name of the new class, and the quoted string that follows is documentation of the class. The remainder of the class definition consists of the definition of four slots. On line 2, the first slot `(is-a :THING)` identifies the superclass of the new class, in this case a system-defined class called `:THING` which encompasses every concept in the ontology. On line 3, the second slot `(role concrete)` identifies the role of the new class as concrete, meaning that it may be instantiated. On lines 4–6, the third slot defines the attribute `name`. Being a `single-slot`, it may contain only one value (as opposed to a `multislot` which may contain multiple values). This slot has two facets that constrain values of the slot. The first facet `(type STRING)` constrains the data type of its value to character strings. The second facet `(cardinality 1 1)` is a cardinality specification which prescribes that each instance of `User` must have both at least and at most one (in other words, exactly one) value in this slot. Finally, the slot on lines 7–9 defines another attribute, `email-address`. This slot is similar to the previous one, with following two exceptions: firstly, it is a `multislot`, thus it can contain multiple values; and secondly, its cardinality ranges from zero to a variable (i.e. unlimited) upper bound.

Given this definition of class `User`, instances may be created. An instance of the class corresponding to the one shown in Figure 4.5, is specified below:

```
1  ([User_0217] of User
2     (name "John Smith")
3     (email-address
4        "john.smith@xyz.com"
5        "john@yahoo.com"))
```

Here, on line 1, `[User_0217]` identifies the instance, and the keyword `of` indicates which class this instance instantiates (in this case the class `User`). Lines 2–5 specify slot values for the indicated slots.

Finally, a notation for the specification of general (i.e. non-facet) constraints is needed. Since those constraints need to express a desired state of a given body of data, a declarative notation such as predicate logic is appropriate. The notation used here is PAL, the Protégé Axiom Language that is employed within the Protégé-2000 system. An example of a constraint corresponding to the one mentioned in Section 4.1.3 above, where owners of workspaces must have an email address ending in `xyz.com`, is shown below (it is assumed that the class `Workspace` has been previously defined, and that this class includes a slot `owner` which holds a reference to an instance of the class `User` who owns the workspace):

```
1 (defrange ?ws :FRAME Workspace)
2 (defrange ?em :FRAME User email-address)
3 (forall ?ws
4    (exists ?em
5       (and (email-address (owner ?ws) ?em)
6            (suffix-of "xyz.com" ?em))))
```

The `defrange` function in line 1 defines a variable `?ws`, which ranges over the frame (instance or class) `Workspace`. Similarly, line 2 defines another variable, `?em`, which ranges over the slot `email-address`, a multi-slot of class `User`. The remaining lines state a constraint involving the two previously defined variables `?ws` and `?em`. The constraint starts on line 3 with the universal quantifier `forall` which is applied to variable `?ws`, thereby applying to all instances of the `Workspace` class. On line 4, the existential quantifier `exists` is applied to the variable `?em`, thereby applying to email addresses in instances of class `User`. Lines 5–6 state the conjunction of a condition that must hold for the quantified variables: the `email-address` of the `owner` of a `Workspace` must be equal to the value of `?em`, and that value of `?em` must end in `xyz.com`. When applied to instances of `Workspace` and `User` in the ontology, the constraint will evaluate to either true or false; it will be true if and only if all instances of `Workspace` are owned by users whose email address ends in `xyz.com`, and will be false otherwise. Using this constraint language, arbitrarily complex constraints can be specified.

4.2 Ontology Specification

The ontology for a given collaboration system consists, as described above, of two parts: on the one hand, a set of models for each layer of the Information Pyramid; on the other hand, a set of mappings between these models. These are specified in a bottom-up fashion according to the modeling method outlined earlier. Here, further details of the specification of these two parts are given.

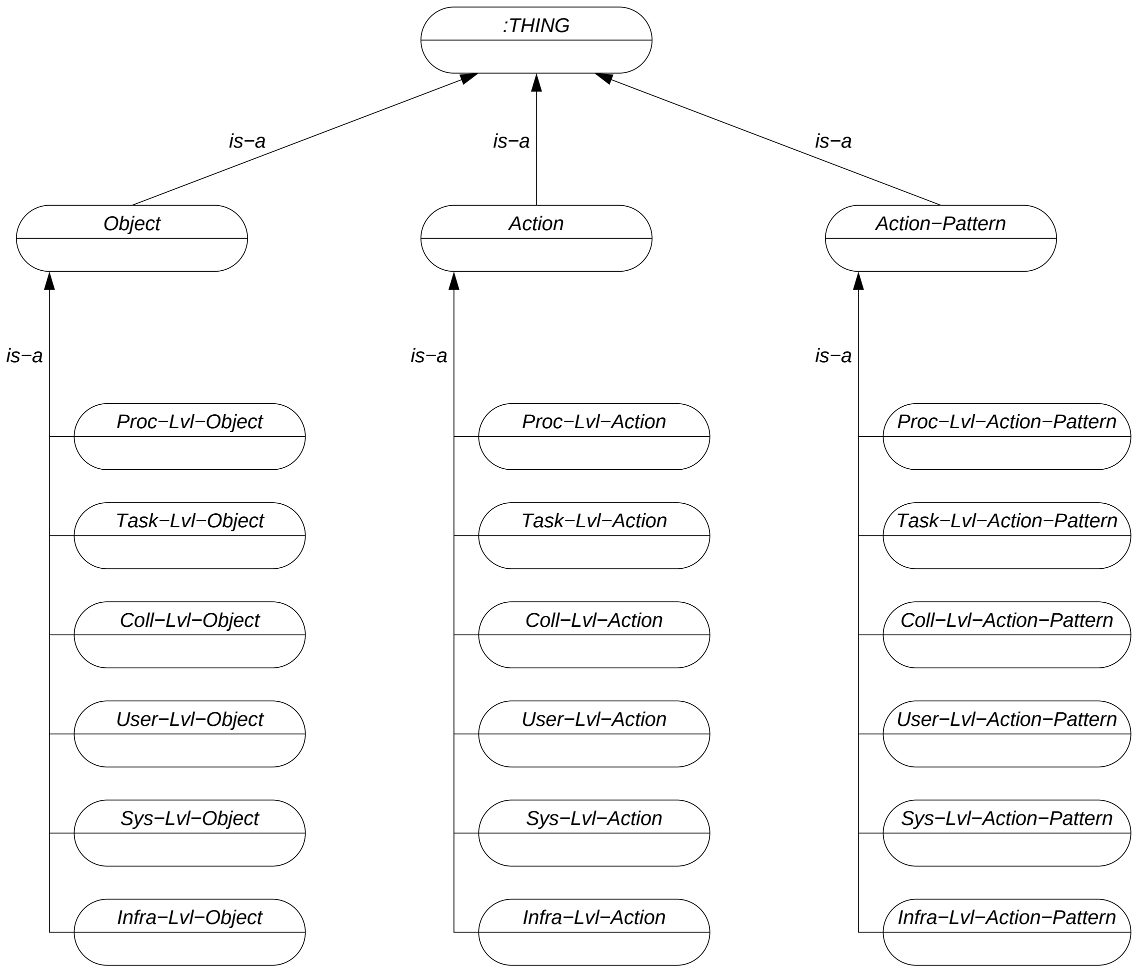


Figure 4.6: Taxonomic hierarchy of common classes of the ontology of collaboration systems

4.2.1 Common Classes

The previous chapter has given a definition of the basic units of information from collaboration systems which are related to the derivation of patterns of virtual collaboration. These are: objects, actions, and action patterns. Furthermore, these basic units of information exist at each level of the Information Pyramid. Thus, a basic structure of *common classes* needed by the ontology of any collaboration system can be defined. These are defined as follows: a set of abstract classes defines the basic units of information themselves, and a set of abstract subclasses of these classes defines units of information for specific levels of the Information Pyramid.

Figure 4.6 shows the taxonomic hierarchy of common classes of the resulting ontology. Note that class names are printed in italics to signify that the classes are abstract, that is, they are not intended to be directly instantiated. The three most fundamental concepts of the ontology are Object, Action, and Action-Pattern (each being a subclass of the ontology's root concept :THING). Each of these has six subclasses, one for each level of the Information Pyramid. Classes corresponding to concepts of a specific collaboration system are then subclasses of these classes.

For instance, the concept User, representing a collaboration system's user, specified on the system level of a given collaboration system would be a subclass of the Sys-Lvl-Object class, since it is an object and it resides on the system level. The relevant class specifications for this concept, as well as for its superclass Sys-Lvl-Object and for that class's superclass Object, are shown below:

```
(defclass Object
   (is-a :THING)
   (role abstract))

(defclass Sys-Lvl-Object
   (is-a Object)
   (role abstract))

(defclass User "A user of a collaboration system."
   (is-a Sys-Lvl-Object)
   (role concrete)
   (single-slot name
      (type STRING)
      (cardinality 1 1))
   (multislot email-address
      (type STRING)
      (cardinality 0 ?VARIABLE)))
```

It can be seen that the classes `Object` and `Sys-Lvl-Object` are defined as abstract, while the class `User` is defined as concrete. The part of the taxonomic hierarchy of classes involved here (and also including the ontology's root class `:THING`) is shown in Figure 4.7.

Other classes defining all of a given collaboration system's objects, actions, and action patterns are specified in a similar manner.

4.2.2 Sessions

As discussed in the previous chapter, collections of actions occurring on a given level of a collaboration system's Information Pyramid can often be mapped to a single action on

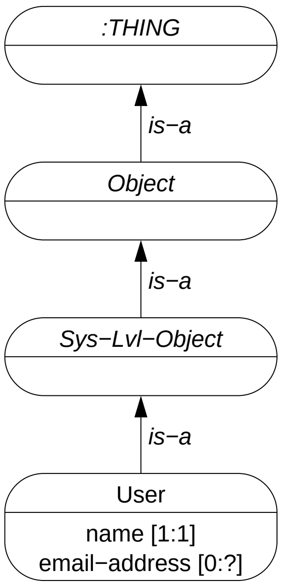


Figure 4.7: Taxonomic hierarchy of four levels of classes in a given collaboration system's ontology

the next-higher level of the Information Pyramid. One example of this is the mapping of multiple instances of the *Post-Discussion-Statement* and *Open-Discussion-Statement* action patterns to the single action pattern *Group-Discussion*, shown in Figure 3.10 on page 84. Mappings such as these are possible when the lower level's instances of action patterns share certain parts of their action context, such as subject (which user or role performed the action), referent (on which object the action was carried out), or location (in which collaboration space the action took place).

However, in some cases of mappings between action patterns, these parts of an action pattern's action context are not sufficient, and the additional context information of the action pattern's *session* is required. A session is defined as follows:

Definition 18 *A **session** is a sequence of actions performed by the same user over a given period of time, with a defined starting and ending point.*
□

Starting and ending points are actions or events that occur in the collaboration system and that it can recognize as delimiting the start and end of a session. These are system-dependent, but common ones include the following:

- **Login/logout:** In some collaboration systems, users login in order to use the system, and logout when finished. In this case, the login and logout actions act as delimiters of a session. That is, the sequence of actions starting with a login action and including all actions until the following logout action constitute a given session in this type of collaboration system.

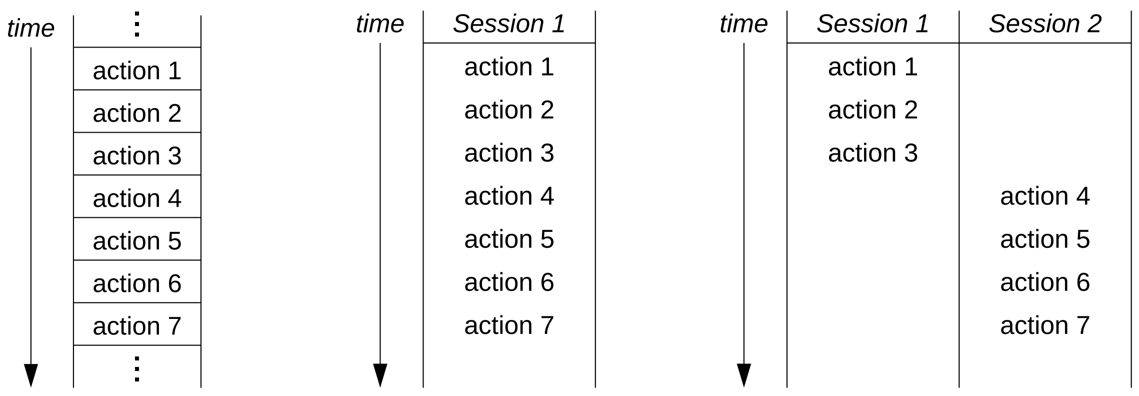
- **Connect/disconnect:** Some collaboration systems do not require user login, and may instead simply serve requests from any connecting client software. In this case, when the client software connects to the server this marks the start of a session, and when it disconnects from the server it marks the session's end. The sequence of actions occurring between the client software's connection and disconnection then constitute a given session in this type of collaboration system.

- **Timeout:** Some collaboration systems do not maintain their connection with the client software from one action to the next, and instead open and close a new connection for each action. In this case, the starting and ending of a session may not be marked by specific events, but instead the end of a session is marked after a specific time period during which no action has been performed, i.e. after a timeout period[2]. The sequence of actions occurring between one timeout and the following timeout then constitute a given session in this type of collaboration system.

To give an example of a mapping of action patterns in which it is necessary to consider the session which the action patterns belong to recall the sequence of the three consecutive system-level action patterns *get_block_tree*, *add_statement*, and *get_block_tree* (shown earlier in Figure 3.8 on page 80). This sequence maps to a single user-level action pattern *Post-Discussion-Statement* (shown earlier in Figure 3.9 on page 82). This mapping, however, is only valid if the three system-level action patterns take place in the same session. Examples of two sequences of action patterns where only the first one can be mapped are shown in Figure 4.8.

Part (a) of the figure shows an extract of a sequence of actions recorded by a collaboration system. Part (b) of the figure shows a case where all the seven actions shown belong to one session (Session 1). Finally, part (c) of the figure shows a different case where only the first three actions belong to one session (Session 1), and the remaining four actions belong to a separate session (Session 2). Assume that actions 3, 4, and 5 are the consecutive system-level actions *get_block_tree*, *add_statement*, and *get_block_tree*. In the case shown in part (b) of the figure these three actions occur in the same session, and it is therefore possible to map the sequence of these three actions to the user-level action pattern *Post-Discussion-Statement*. On the other other hand, in the case shown in

[2]Such a timeout is system-defined and thus varies from one system to another; however, timeout values of around 10–30 minutes are common.



(a) Sequence of actions (b) Actions belonging to the same session (c) Actions belonging to separate sessions

Figure 4.8: A sequence of actions belonging to one or more sessions

part (c) of the figure this is not possible, because the three actions do not belong to the same, but to two different sessions. The session thus extends the action context of the involved actions, and constitutes part of an action's location: a *logical* location.

Sessions are modeled as special kinds of objects. Sessions are composite objects consisting of some general information about the session itself, and a set of actions that belong to the session. Below is the definition of a class `Session` which models the session and consists of a numeric session identifier and a set of actions belonging to the session:

```
(defclass Session
   (is-a Object)
   (role concrete)
   (single-slot SessionID
      (type INTEGER)
      (range 1 ?VARIABLE)
      (cardinality 1 1))
   (multislot Session-Actions
      (type INSTANCE)
      (allowed-classes Action)
      (cardinality 1 ?VARIABLE)))
```

A diagram of the relevant part of the ontology is shown in Figure 4.9. Here two different kinds of class relationships are shown: the *is-a* relationship where one class

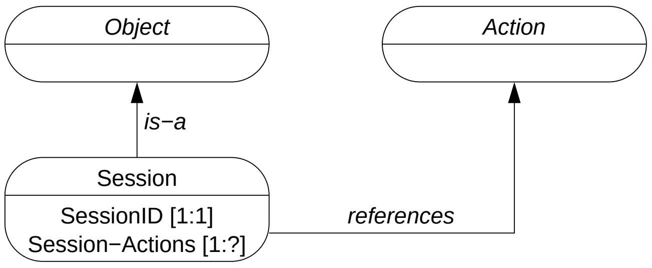


Figure 4.9: Class `Session` for representing sessions in a collaboration system's ontology

subclasses another class; and the *references* relationship where one class references another class through one or more of its slots.

Using this class, an actual session can be represented by creating an instance of `Session` and filling its `Session-Actions` slot appropriately with the actions that make up the session, in the sequence of their occurrence.

4.2.3 Specification of Concept Mappings

Initially, after concepts of the ontology of virtual collaboration have been specified, the result are separate sets of concepts on different levels of the Information Pyramid. However, many of these concepts on different levels are related. To be able to map concepts on a lower level to concepts on the next-higher level, it is necessary to identify how these concepts are related, and then to specify this. However, this specification of the mapping of concepts depends on the type of relationship between the concepts involved. Here, following three main types of relationships are distinguished:

1. Unmodified one-to-one correspondence
2. One-to-one correspondence with modifications
3. Many-to-one correspondence

The specification of mappings for each of these types of relationships is considered next.

4.2.3.1 Unmodified One-To-One Correspondence

Unmodified one-to-one correspondence means that two concepts on adjacent levels are identical in terms of all their slots and facets. This is the simplest of all cases as no explicit mapping is required (or rather, the mapping in this case is the identity function). However, the lower-level concept is specified as a subclass of its level's type of class (e.g. as a system-level object). This marks the class as belonging to that level. In order to indicate that the same class also represents an identical concept on a different level, the class specification needs to be extended by adding the appropriate superclass to its `is-a` slot.

Consider the case of a class representing users of a collaboration system (only the first two lines of the class specification are shown):

```
(defclass User "A user of a collaboration system."
   (is-a Sys-Lvl-Object)
   ...
```

This class specification is for an object on the system level, indicated by its superclass `Sys-Lvl-Object`. If the same concept also exists on the user level in unmodified form, the class specification can simply be extended to the following:

```
(defclass User "A user of a collaboration system."
   (is-a Sys-Lvl-Object User-Lvl-Object)
   ...
```

Here, the `is-a` slot is now extended by a second superclass, `User-Lvl-Object`. The remainder of the class specification, however, remains unchanged. This now indicates that the class `User` is an object that belongs to both the system and user levels, and has an identical definition on both levels.

4.2.3.2 One-To-One Correspondence With Modifications

One-to-one correspondence with modifications means that for a given concept on one level, there exists a corresponding concept on the level above it, but with a different definition of its slots and facets. In this case it is not possible to simply add the upper level's superclass to the `is-a` slot of the lower-level class, as the class definitions are not identical. Instead, there are two separate classes whose correspondence is explicitly expressed in the form of a mapping. This mapping establishes which slot belonging to the lower-level class maps to which slot belonging to the upper-level class.

For instance, consider the two classes `User` and `Person`. The former belongs to the system level and records such information as userid, password and user name, while the latter belongs to the user level and consists only of userid and username, that is, it does not include the password. The two class definitions are shown below:

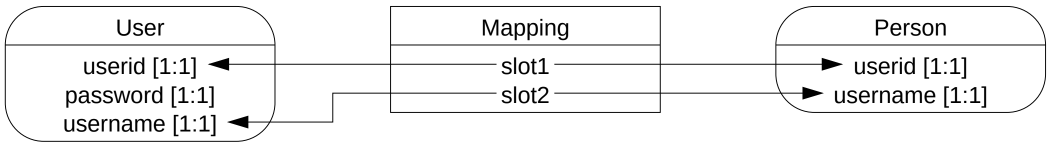


Figure 4.10: Mapping of object class `User` to object class `Person`

```
(defclass User "A user of a collaboration system."
   (is-a Sys-Lvl-Object)
   (role concrete)
   (single-slot userid
      (type STRING)
      (cardinality 1 1))
   (single-slot password
      (type STRING)
      (cardinality 1 1))
   (single-slot username
      (type STRING)
      (cardinality 1 1)))

(defclass Person "A person using a collaboration system."
   (is-a User-Lvl-Object)
   (role concrete)
   (single-slot userid
      (type STRING)
      (cardinality 1 1))
   (single-slot username
      (type STRING)
      (cardinality 1 1)))
```

Here, the slots `userid` and `username` from the `User` class appear in the definition of the `Person` class. What is required is to establish that these slots map across the two classes. In the form of a diagram, the mapping between these two classes is shown in Figure 4.10.

This mapping maps from the `User` class to the `Person` class. In a mapping of classes, that which is being mapped from is called the *source class*, and that which is being mapped to is called the *target class*.

Simple slot mappings

In the figure, a mapping construct between classes `User` and `Person` establishes correspondence of slots. This mapping construct is defined here in the form of special mapping classes. The first one, `Class-Mapping`, represents the mapping to a target class:

```
(defclass Class-Mapping
   (is-a :THING)
   (role concrete)
   (single-slot Target-Class
      (type SYMBOL)
      (allowed-parents :THING)
      (cardinality 1 1))
   (multislot Slot-Map
      (type INSTANCE)
      (allowed-classes Slot-Mapping)
      (cardinality 1 ?VARIABLE)))
```

It identifies the target class that is being mapped to and consists of one or more slot mappings (the source class(es) being mapped from are identified inside the slot mappings). The second mapping class, `Slot-Mapping`, represents the mapping to a slot in the target class. Because there are several different ways that slots can be mapped, the `Slot-Mapping` class is defined as abstract and concrete subclasses are defined for the different kinds of mappings to slots. Below are definitions of the abstract class `Slot-Mapping` and of one subclass, `Simple-Slot-Mapping`, which maps a slot from a source class to a slot in the target class:

```
(defclass Slot-Mapping
   (is-a :THING)
   (role abstract)
   (single-slot Target-Class
      (type SYMBOL)
      (allowed-parents :THING)
      (cardinality 1 1))
   (single-slot Target-Slot
      (type INSTANCE)
      (allowed-classes :STANDARD-SLOT)
      (cardinality 1 1)))
```

```
(defclass Simple-Slot-Mapping
   (is-a Slot-Mapping)
   (role concrete)
   (single-slot Source-Class
      (type SYMBOL)
      (allowed-parents :THING)
      (cardinality 1 1))
   (single-slot Source-Slot
      (type INSTANCE)
      (allowed-classes :STANDARD-SLOT)
      (cardinality 1 1)))
```

The slots holding source and target classes in these two class definitions are constrained to the class `:THING`, meaning that any classes can be mapped (since `:THING` is the root of the class hierarchy). Similarly, the source and target slots are constrained to instances of `:STANDARD-SLOT`, which is the superclass of all slots in the ontology, thereby allowing any slot to be mapped. Because `Simple-Slot-Mapping` is a subclass of `Slot-Mapping`, it inherits the two slots `Target-Class` and `Target-Slot` from its parent class.

Using the `Class-Mapping` and `Simple-Slot-Mapping` classes as mapping tools, it is now possible to specify mappings of slots for any corresponding classes across levels of the Information Pyramid. This is done by creating instances of the mapping classes and setting slot values appropriately.

Aggregated slot mappings

In some cases it may not be sufficient to simply map slots from one class to another, that is, to map a single slot value from a single instance of a source class to a slot in a target class. Instead, sometimes an *aggregation* of slot values from instances of a source class to a slot in a target class is required.

Consider the example of a collection of versioned documents where each member of the document collection constitutes a different version of the same document. On one level, say the user level, the class `Document` may represent an individual document in the collection, while on the next-higher level, the collaboration level, the class `Versioned-Document` may represent the collection of all individual documents that make up the versioned document. In this case, the class `Document` may be mapped to the class `Versioned-Document`, this mapping too being of the type “one-to-one correspondence with modifications”. That is, when considering the mapping on the level of classes,

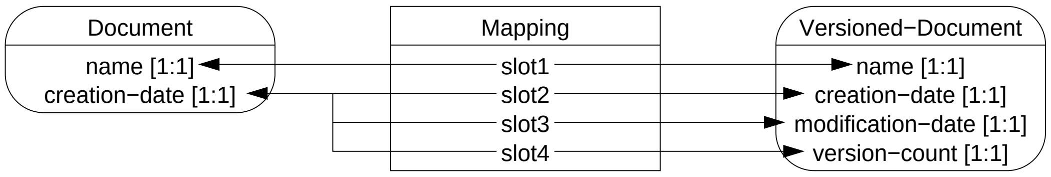


Figure 4.11: Mapping of object class `Document` to object class `Versioned-Document`

a single source class is mapped to a single target class. This is shown in Figure 4.11.

When considering this mapping on the level of instances however, multiple instances of the source class are mapped to a single instance of the target class. Here, values of the source class's instances are *aggregated* into a single value of a slot in the target class. In the example given here, the value of the target slot `creation-date` represents the date of the creation of the first version of the document. It can be obtained by getting the smallest value of the slot `creation-date` from all corresponding instances of `Document`. On the other hand, the value of the target slot `modification-date` represents the date of the latest modification of the versioned document, which is the date of the creation of the latest version of the document. This can be obtained by getting the largest value of the slot `creation-date` from all corresponding instances of `Document`. Finally, the target slot `version-count` holds the number of versions of the document collection. It can be obtained by counting the number of instances of the source class, or by counting the number of values of a given slot in the source class. In the example, a count of all values of the slot `creation-date` yields the count of versions of the document.

Specifying mappings of slots which are obtained through aggregation requires the specification of an *aggregation function* for the source slot. A number of such aggregation functions can be defined, including the following:

- **all:** given a set of values, returns the whole set of all values
- **any:** given a set of values, returns any one of the values
- **avg:** given a set of numerical values, returns the average, i.e. the arithmetic mean, of those values
- **count:** given a set of values, returns a count of the values, i.e. the cardinality of the set
- **max:** given a set of values, returns the largest value
- **min:** given a set of values, returns the smallest value

- **sum:** given a set of numerical values, returns the sum of all values

Another subclass of the `Slot-Mapping` class is defined here for the specification of aggregated slots, namely the class `Aggregated-Slot-Mapping` shown below:

```
(defclass Aggregated-Slot-Mapping
   (is-a Slot-Mapping)
   (role concrete)
   (single-slot Source-Class
      (type SYMBOL)
      (allowed-parents :THING)
      (cardinality 1 1))
   (single-slot Source-Slot
      (type INSTANCE)
      (allowed-classes :STANDARD-SLOT)
      (cardinality 1 1))
   (single-slot Aggregation-Function
      (type SYMBOL)
      (allowed-values all any avg count max min sum)
      (cardinality 1 1)))
```

Since this class is a subclass of the class `Slot-Mapping`, it inherits its slots `Target-Class` and `Target-Slot`. One of its own slots, `Aggregation-Function`, identifies the aggregation function to be performed on all instances' source slot. Allowed values for this slot are the seven functions mentioned above, but this facet could be extended to allow for other aggregation functions as needed.

4.2.3.3 Many-To-One Correspondence

Many-to-one correspondence means that two or more concepts on one level correspond to a single concept on the next-higher level. In this case, the slots of the upper-level class originate from the lower-level classes. Again, this needs to be expressed in the form of a mapping.

An example of this is shown in Figure 4.12. This example concerns a shared white-board for writing and drawing. On one level this is represented as an instance of the class `Board` for each whiteboard, which contains references to one or more instances of `Page`, the pages of the whiteboard. On the next-higher level, however, the whiteboard is represented by a single class, `Whiteboard`, which contains the information that is relevant at that level, in this case the name of the board and a list of users who have drawn on it (i.e.

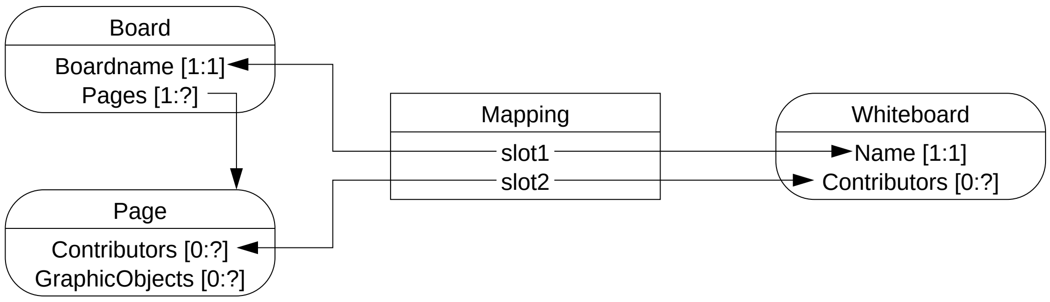


Figure 4.12: Mapping of object classes `Board` and `Page` to object class `Whiteboard`

its contributors). The first attribute, `Name` is mapped from the slot `Boardname` of class `Board` (note that the mapping of slots does not require the source and target slots to have the same names). The second attribute, `Contributors`, is mapped from the slot with the same name in the class `Page`.

To realize these mappings once again requires the specification of mappings of the class and its slots, for which the mapping classes `Class-Mapping` and `Simple-Slot-Mapping`, defined above, are used. That is, the mapping of concepts for the case of one-to-one correspondence with modifications, and the case of many-to-one correspondence is achieved in the same manner. However, in order to achieve a valid mapping in the case of many-to-one correspondence, the specified mapping needs to be *constrained*, to ensure that only pages which are part of the whiteboard being mapped are included in the mapping. This *mapping constraint* is expressed in Figure 4.12 through the arrow from slot `Pages` in class `Board` to class `Page`. When writing functions to map instances of concepts across levels, this constraint needs to be considered. This is discussed below in Section 4.2.4.

Instance mappings

The examples of mapping slots across classes shown above assumed that the target slot of a target class can be mapped from a specific source slot of a source class. However, this may not always be the case. In some cases, for instance, a class may exist at one level, but may not be referenced in any slot of any class on that level. If an instance of this class has to be mapped to a target slot in a class on the next-higher level, the mapping classes shown above can not be used, as they require any instance that is to be mapped to exist as the value of a slot in some class (i.e. the instance is referenced by that slot). Instead, another kind of mapping is needed that takes an instance of a source class and directly maps it to a target slot in a target class.

To illustrate this, consider the case of an object `Brainstorming-Room` which repre-

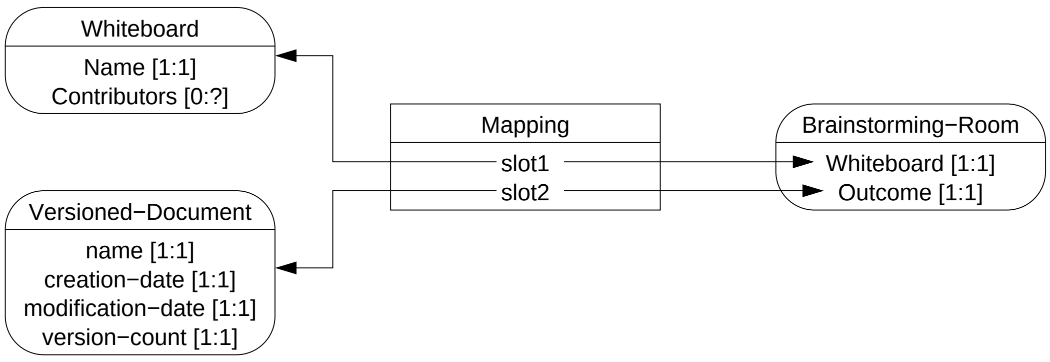


Figure 4.13: Mapping of instances of object classes `Whiteboard` and `Versioned-Document` to object class `Brainstorming-Room`

sents a virtual "room" for brainstorming. This "room" may be equipped with an electronic whiteboard, which is an instance of object class `Whiteboard`, on which people in the room write or draw their ideas; as well as with a versioned document, which is an instance of object class `Versioned-Document`, to record the outcomes of the brainstorming that takes place in the virtual room, including its history, for which multiple versions of the document are maintained. Given that no class on this level contains references to instances of the two classes `Whiteboard` and `Versioned-Document`, a mapping to class `Brainstorming-Room` requires entire instances to be mapped. This example is illustrated in Figure 4.13. Here, the first slot of `Brainstorming-Room`, the slot `Whiteboard`, is mapped from an instance of the class with the same name (`Whiteboard`), while the second slot, `Outcome`, is mapped from an instance of class `Versioned-Document`.

To specify this kind of mapping requires a different kind of mapping class. For this purpose the mapping class `Instance-Slot-Mapping` is defined:

```
(defclass Instance-Slot-Mapping
   (is-a Slot-Mapping)
   (role concrete)
   (single-slot Source-Instance
      (type INSTANCE)
      (allowed-classes :THING)
      (cardinality 1 1)))
```

This mapping class is a subclass of `Slot-Mapping` and thus inherits its slots `Target-Class` and `Target-Slot`. In addition, it contains the slot `Source-Instance` which holds a reference to the instance to be mapped. This slot is constrained to instances of the class `:THING`, meaning that any class can be mapped.

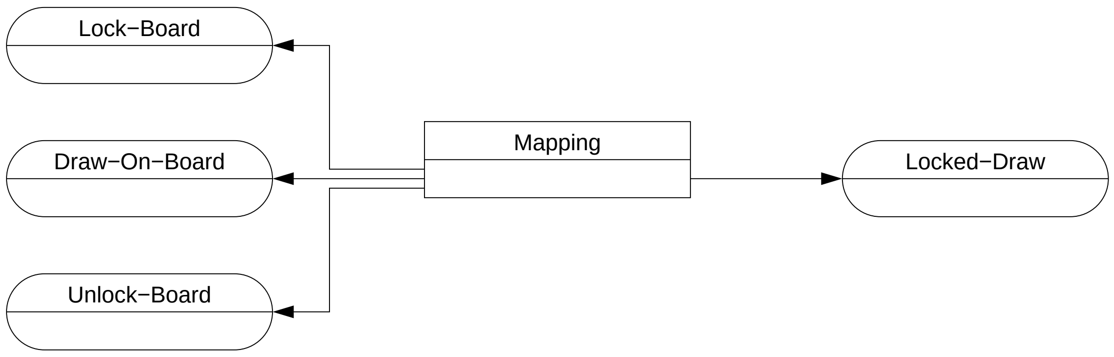


Figure 4.14: Mapping of sequence of action classes `Lock-Board`, `Draw-On-Board`, and `Unlock-Board` to action class `Locked-Draw`

Action/action pattern sequence mappings

The case of mapping of multiple actions (and action patterns) is another special case that has to be considered. Just as with objects, multiple actions and action patterns on one level can also correspond to ones on the next-higher level. Thus the correspondence of slots can be specified in the same manner as was shown for objects above. However, actions occur in a temporal sequence, and this sequence may be of importance in determining correspondence of actions across levels. For instance, the sequence of actions `Lock-Board`, `Draw-On-Board`, and `Unlock-Board` may occur on one level of the Information Pyramid whenever a user performs a drawing operation on a shared whiteboard with concurrency control implemented through locking. This specific sequence of actions may correspond to the single action `Locked-Draw` on the next-higher level, as shown in Figure 4.14. However, if the sequence of actions on the lower level were changed, the correspondence would no longer be valid. Thus for correspondences of multiple actions on one level to a single action on the next-higher level, it is important to not only specify mappings of slots, but also the specific sequence of actions.

Action (and action pattern) sequences can be expressed using another mapping construct, the class `Sequence-Mapping`, the definition of which is shown below:

```
(defclass Sequence-Mapping
   (is-a :THING)
   (role concrete)
   (single-slot Mapping-Target
      (type SYMBOL)
      (allowed-parents Action Action-Pattern)
      (cardinality 1 1))
   (multislot Sequence-Elements
      (type SYMBOL)
      (allowed-parents Action Action-Pattern)
      (cardinality 1 ?VARIABLE)))
```

Here, the slot `Mapping-Target` identifies the target action or action pattern of the mapping, such as the class `Locked-Draw` mentioned in the example above. The multislot `Sequence-Elements` holds the classes of actions or action patterns, in the sequence in which they are to be mapped to the target. Thus for the `Locked-Draw` action target, the `Sequence-Elements` slot would contain the three action classes `Lock-Board`, `Draw-On-Board`, and `Unlock-Board`, in that sequence.

Given on the one hand specifications of action sequences, expressed through instances of `Sequence-Mapping`, and on the other hand specifications of sessions containing actions in the sequence in which they actually occurred, through instances of `Session`, makes it possible to identify higher-level actions within sessions. This involves the matching of action sequences in the session with action sequence mappings.

———— . ————

To summarize, this section has proposed the mapping construct `Class-Mapping` for the specification of mappings of classes from one level to the next-higher level; together with following four mapping constructs for the mapping of slots and instances from one level to the next-higher level:

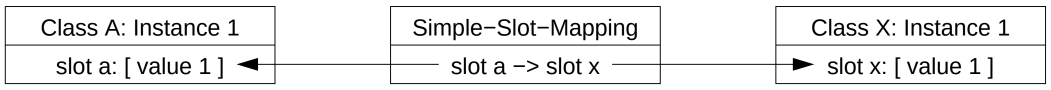
1. `Simple-Slot-Mapping`: Map one slot from one instance of one class to one slot in another class; applicable to both objects and actions/action patterns.

2. `Aggregated-Slot-Mapping`: Map one slot from multiple instances of one class to one slot in another class, performing some aggregation on the multiple slot values; applicable to both objects and actions/action patterns.

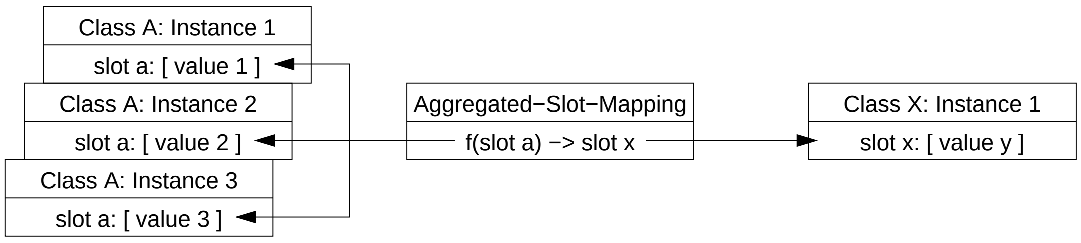
3. `Instance-Slot-Mapping`: Map an instance of one class to a slot in another class; applicable to both objects and actions/action patterns.

4. `Sequence-Mapping`: Map a sequence of instances of classes to an instance of another class; applicable to actions/action patterns only.

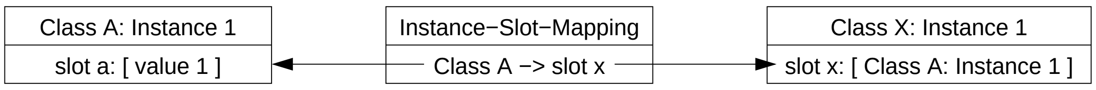
Illustrations of these four mapping constructs are shown in Figure 4.15. For each of the four example mappings, the left-hand side shows one or more instances of classes to



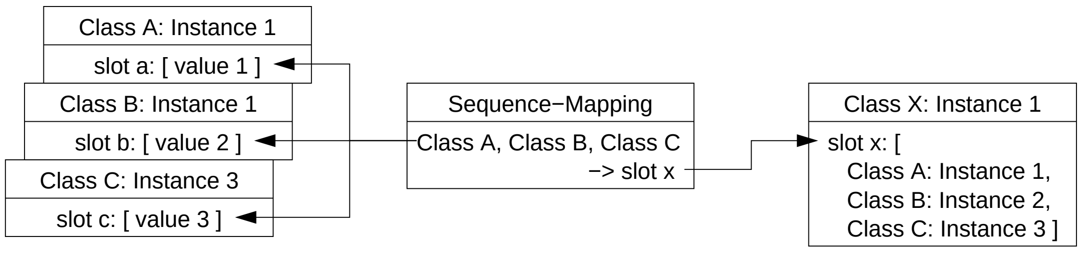
(a) `Simple-Slot-Mapping`: map one slot of one instance of one class to one slot of another class



(b) `Aggregated-Slot-Mapping`: map one slot of multiple instances of one class to one slot of another class



(c) `Instance-Slot-Mapping`: map one instance of one class to one slot of another class



(d) `Sequence-Mapping`: map a sequence of instances of classes to an instance of another class

Figure 4.15: Mapping constructs used for mapping slots and instances across levels

be mapped, the middle shows the mapping construct, and the right-hand side shows an instance of the resulting mapped class.

The list of mapping constructs proposed above is for some of the more obvious types of class mappings, rather than being a complete list of all possible class mapping constructs. However, when other types of class mappings are needed, other mapping constructs can be defined in a similar manner as the ones above. Moreover, certain types of mappings may be subject to constraints that need to be satisfied. In this case, the specification of mappings needs to include these constraints. This is discussed further in the following section, and is illustrated in Chapter 5.

4.2.4 Definition of Mapping Functions

The specification of concept mappings above has identified how concepts on one level of the Information Pyramid map onto concepts on the next-higher level. Given this information about mappings, it is now possible to map actual *instances* of these concepts across levels. Doing so involves taking instances of those classes which appear as source classes of a given class mapping, retrieving slot values according to the specified slot mappings and subject to any identified constraints, and using these to construct new instances of the target class. In order to facilitate this mapping of instances, *mapping functions* are defined.

A mapping function is a function that creates instances of a specific target class. The function receives input parameters holding references to instances of the concept mapping's source class(es), and produces instances of the target class. An example of a mapping function, `create-whiteboard`, for the mapping of instances of `Board` and `Page` to `Whiteboard` is shown below. In pseudocode this function is defined as follows[3]:

```
FUNCTION create-whiteboard (board)
  IF board is an instance of class Board THEN
    initialize the list of contributors to an empty list;
    FOR all pages of board DO
      add the contributors of the page to the list of contributors;
    END_FOR
    create an instance of class Whiteboard using the value of
      Boardname of board and the list of contributors;
  END_IF
END_FUNCTION
```

[3]The pseudocode notation employed here and in the rest of this chapter is based on the Program Design Language of (Easteal and Davies, 1989).

For use in the Protégé-2000 system this function can be implemented using the function `deffunction`, as follows[4]:

```
(deffunction create-whiteboard (?board)
  (if (and (instancep ?board)
           (eq (class ?board) Board))
      then
    (bind $?contrib (create$))
    (foreach ?page (slot-get ?board Pages)
             (bind $?contrib
                   (union$ $?contrib (slot-get ?page Contributors))))
    (make-instance of Whiteboard
                   (Name (slot-get ?board Boardname))
                   (Contributors $?contrib))))
```

This function, `create-whiteboard`, takes one input parameter `?board` which holds a reference to the instance of `Board` to be mapped. It then performs input validation, testing whether `?board` is actually an instance (using predicate `instancep`), and whether its class is `Board` (using predicate `class` to obtain the instance's class, then testing for equality with predicate `eq`). If these conditions are satisfied, an instance of `Whiteboard` can be created. First the set of contributors is obtained by iterating over all instances of `Page` held in the slot `Pages` of `Board` (retrieved using the function `slot-get` which returns the value of a given slot), and retrieving the value of the slot `Contributors` of each instance of `Page`. These sets of values are joined together using the set union function `union$`, then bound to the variable `$?contrib`. Finally, the new instance of `Whiteboard` is created using function `make-instance`, with the name taken from the slot `Boardname` of `Board`, and the set of contributors being the one previously constructed and stored in variable `$?contrib`.

The function thus implements the mapping of the two lower-level classes `Board` and `Page` to the upper-level class `Whiteboard` corresponding to the specification of its class and slot mappings. The constraint identified earlier, namely that instances of `Page` to be mapped need to be elements of the multi-slot `Pages` of the instance of `Board` involved in the mapping, is implicitly satisfied, as the mapping function only involves a single input parameter referencing the instance of `Board`, and any instances of `Page` are obtained through the slot `Pages` of that instance of `Board`.

[4]This and all following functions are defined in the language of Jess (Friedman-Hill, 2001), an expert system shell that has the ability to manipulate ontologies in the Protégé-2000 system through the JessTab software (Eriksson, 2001).

Once defined, mapping functions are added to, and become an integral part of, a collaboration system's ontology.

4.3 Extraction of Patterns of Virtual Collaboration

After the ontology of virtual collaboration for a given collaboration system has been specified, including specifications of concepts and concept mappings as well as mapping functions, patterns of virtual collaboration can be extracted from data collected by that collaboration system. Two types of pattern extraction can be distinguished: extraction of patterns directly from the data collected by the collaboration system, and extraction of patterns through derivation from other patterns. Each of these is detailed below.

4.3.1 Base Level Pattern Extraction

Data at the base level of the Information Pyramid of Virtual Collaboration forms the source for the extraction of the lowest-level patterns of virtual collaboration. This data is of two types:

1. Data on objects maintained by the collaboration system.
2. Data on actions performed in the collaboration system.

As discussed in Section 3.2, the former is commonly stored in files or database tables, while the latter is typically stored in the form of some type of server log file that records actions that have been performed.

The extraction of action patterns initially requires that the source data on objects and actions recorded by the collaboration system be transformed into a form that is amenable to subsequent pattern extraction. Each collaboration system typically has its own unique format in which it stores records of objects and actions. For each object record, an instance of the corresponding object class (as specified in the ontology) is created, which then can be added to the ontology. The same applies to action records.

In pseudocode, the creation of instances of object classes is expressed as follows:

```
FOR all object records DO
  lookup ontology definition of corresponding object class;
  create instance of object class with values from object record;
  add object instance to ontology;
END_FOR
```

Instances of action classes are created in a similar fashion, with details obtained from records of actions. As a result, representations of objects and actions in the format of the ontology of virtual collaboration exist. This then makes the extraction of action patterns possible. For each action instance, a corresponding action pattern instance is created. As instances of action pattern classes contain details that are also obtained from action records, for example attributes that belong to an action's action context, the extraction of action pattern instances is performed together with the creation of action instances. This is shown in the following pseudocode:

```
FOR all action records DO
  lookup ontology definition of corresponding action class;
  create instance of action class with values from action record;
  add action instance to ontology;

  lookup ontology definition of corresponding action pattern class;
  identify related object instances from action record;
  create action pattern instance referencing action instance and
    identified object instances;
  add action pattern instance to ontology;
END_FOR
```

The entire extraction process is illustrated in Figure 4.16. At the bottom of the figure the data sources are shown, namely records of objects and actions in the format of the collaboration system from which they were obtained. The upper part of the figure shows the ontology of virtual collaboration, containing classes and instances of objects, actions, and action patterns. The data sources are transformed into instances of objects and actions (indicated by the left and right thick upward-pointing arrows). These instances become part of the ontology of virtual collaboration, being instances of the corresponding object and action classes, respectively. Finally, instances of action patterns are extracted from actions, objects, and information contained in action records (indicated by the three thick arrows pointing to the central section of the ontology).

For instance, to continue the whiteboard example given above in Section 4.2.3.3, a collaboration system's object records may include records of whiteboards and their pages, while action records may include records of actions performed on the whiteboards' pages. To extract action patterns from these records, firstly instances of the objects are created and added to the ontology, in this case instances of the object classes `Board` and `Page`. This is followed by creation of instances of the action classes, such as the `Lock-Board`, `Draw-On-Board`, and `Unlock-Board` action classes shown earlier

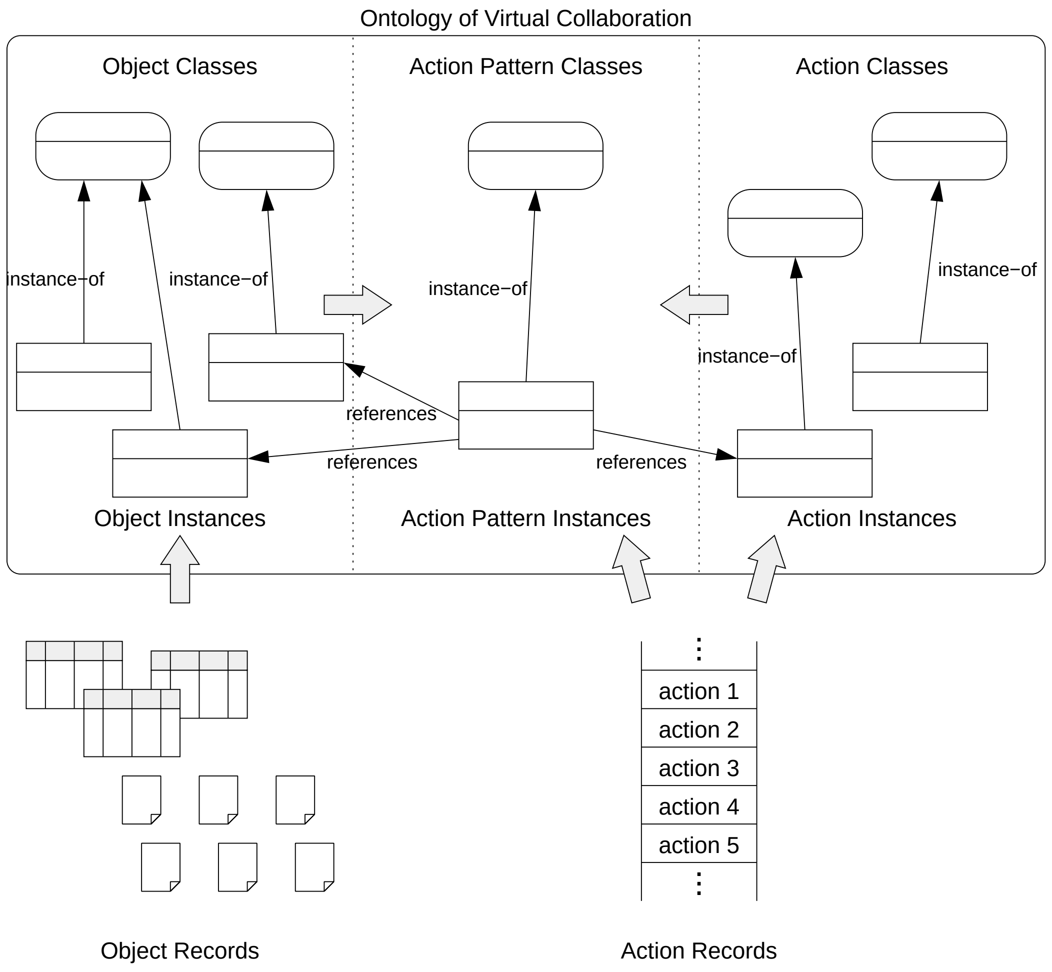


Figure 4.16: Extraction of instances of action patterns from object and action records at the collaboration system's base level, via object and action instances

(cf. Figure 4.14). Finally, for each instance of these action classes, an instance of a corresponding action pattern class is produced. Thus an instance of action pattern class `Lock-Board-Pattern` (an action pattern corresponding to the `Lock-Board` action) is created from instances of the object `Board` and the action `Lock-Board`, as well as action context information obtained from the collaboration system's action record. In like manner, instances of other action patterns are produced.

Once all object and action records have been processed, and instances of the corresponding object and action classes have been created, and instances of action pattern classes have been extracted from these, the extraction of base-level action patterns is complete. This is followed by the extraction of higher-level action patterns, detailed next.

4.3.2 Higher-Level Pattern Extraction

Patterns on higher levels of the Information Pyramid (i.e. higher than the base level) are extracted by exclusively using classes and instances in the ontology of virtual collaboration, without having to refer to object and action records of the collaboration system. That is, instances of higher-level object, action, and action-pattern classes are created from corresponding lower-level instances in the ontology. The process of pattern extraction resembles that at the base level: first object instances are created, next action instances are created, and finally action pattern instances are extracted from object and action instances. For each of these, i.e. objects, actions, and action patterns, instances are created through (previously defined) mapping functions.

Object instances for a given level are created by iterating all object mapping functions over all possible input objects on the level below. Thus if an object mapping function creates instances of an object class, say X, from instances of another object class, say Y, the object mapping function is invoked for each such possible instance of object class Y. The resulting instances of object class X are then added to the ontology. This is shown in the pseudocode below:

```
FOR all object mapping functions DO
  FOR all possible input object instances DO
    call mapping function to create instance of target object class;
    add object instance to ontology;
  END_FOR
END_FOR
```

Instances of actions are created in a similar fashion: action mapping functions iterate over all possible input actions to create instances of action classes on a given level. This is shown in the pseudocode below:

```
FOR all action mapping functions DO
  FOR all possible input action instances DO
    call mapping function to create instance of target action class;
    add action instance to ontology;
  END_FOR
END_FOR
```

Finally, instances of action patterns are extracted. These reference previously created instances of objects, actions, and action patterns. This is shown in the pseudocode below:

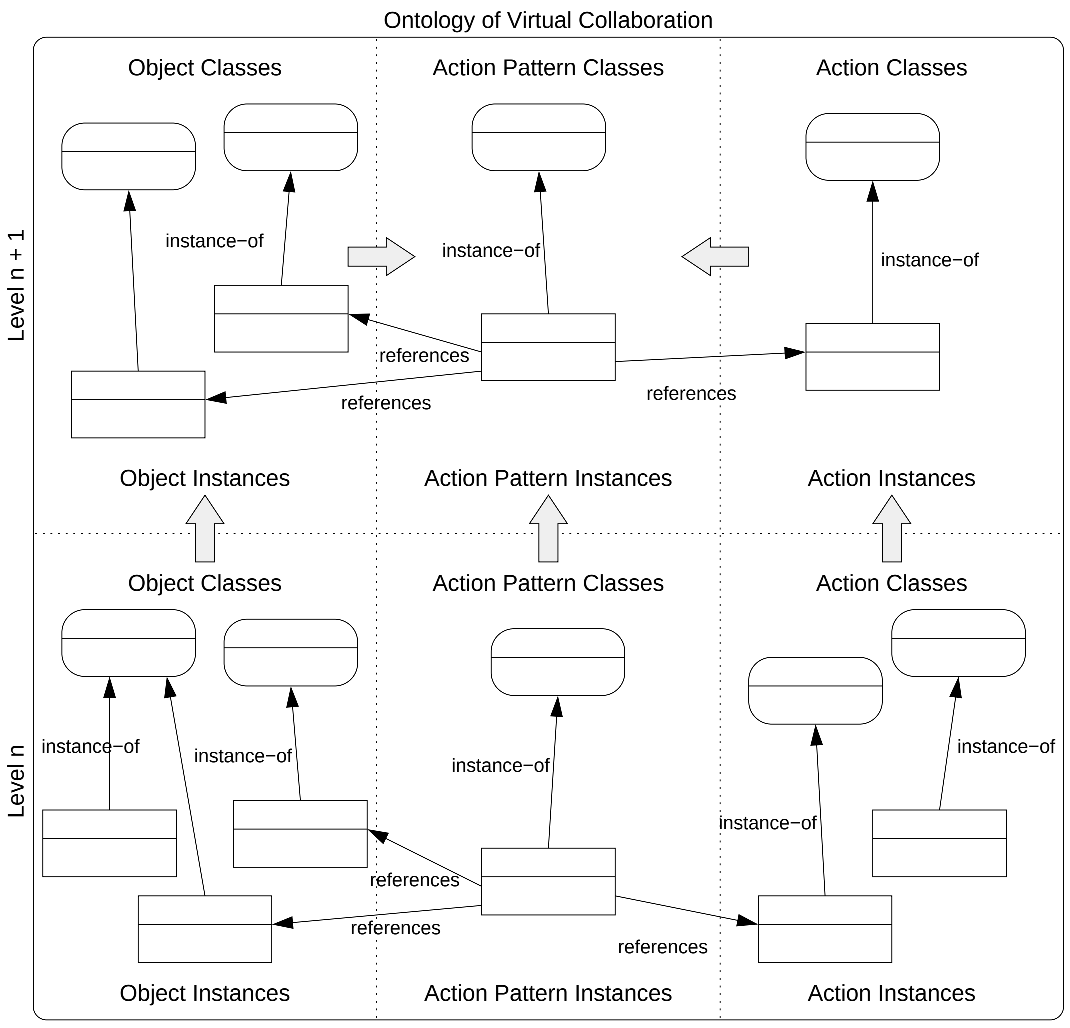


Figure 4.17: Extraction of instances of level $n+1$ action patterns from level $n+1$ objects and actions and level n action patterns

```
FOR all action pattern mapping functions DO
  FOR all possible input action pattern instances DO
    call mapping function to create instance of target action pattern
      class;
    add action pattern instance to ontology;
  END_FOR
END_FOR
```

This extraction process is illustrated in Figure 4.17. It shows the ontology of virtual collaboration, including two adjacent levels (the upper and lower half of the box). At each level there are classes and instances of objects, actions, and action patterns. The thick arrows represent the mapping of instances of these classes from one level to the level above (performed, as mentioned above, by mapping functions), and finally the extraction of instances of action patterns from instances of objects and actions on the same level, as well as from instances of action patterns on the level below.

To continue the above whiteboard example, instances of the `Whiteboard` object class (which in the ontology of Figure 4.17 would reside at level $n+1$) are created from instances of the `Board` object class (which in the ontology of Figure 4.17 would reside at level n) by applying the mapping function `create-whiteboard` (shown on page 119 above). Other mapping functions are used to create instances of level $n+1$ action classes from their level n source classes. For example, an instance of (level $n+1$) action class `Locked-Draw` is created from the sequence of (level n) actions `Lock-Board`, `Draw-On-Board`, and `Unlock-Board` (corresponding to the mapping shown in Figure 4.14 above). Finally, instances of action patterns are created, referencing related instances of object and action classes. Thus an instance of the (level $n+1$) `Locked-Draw-Pattern` action pattern class (an action pattern corresponding to the `Locked-Draw` action) is created from instances of the (level $n+1$) `Whiteboard` object class and `Locked-Draw` action class, as well as action context information obtained from instances of the (level n) action pattern instances `Lock-Board-Pattern`, `Draw-On-Board-Pattern`, and `Unlock-Board-Pattern`. Instances of other action pattern classes are created in a corresponding manner.

The mapping of object and action instances, and the extraction of action pattern instances, continues one-by-one on all levels of the Information Pyramid above the base level, until it finally reaches the highest level. The result is an ontology that is complete with classes and instances of all objects, actions, and action patterns across all levels.

4.4 Visualization of Patterns of Virtual Collaboration

The modeling method for producing models and mappings of different levels of the Information Pyramid that was proposed earlier in this chapter presented a bottom-up approach for the derivation of patterns of virtual collaboration. This approach takes fine-grained low-level information and transforms it into successively more high-level representations of collaborative activity. On the highest level, the process level, combinations of collaboration spaces specially configured to support specific processes are identified.

One way in which the modeling of action patterns differs across the different levels is in the *source* from which patterns are obtained. Two main sources can be distinguished: on the one hand the *collaboration system* itself, on the other hand the collaboration system's *users*.

The lowest-level patterns, such as those from the Information Pyramid's micro level, relate to events generated by the system at that level. Therefore, the source of the patterns is entirely the collaboration system itself. Thus by inspecting the set of commands that the collaboration system makes available, patterns involving those commands are identified and specified, without having to observe the system in use.

The highest-level patterns, on the other hand, such as those at the task and process levels, relate to tasks and processes performed by the collaboration system's users. The source of these patterns is necessarily the group of users of the system who perform these tasks and processes. In order to be able to identify and specify patterns at these levels, it is necessary to refer to records of actual tasks and processes that have been performed through the system.

Finally, on intermediate levels the source of patterns may consist of a combination of both the collaboration system and its users. For example, the system provides basic objects and actions that can be performed on these objects, while the users provide specific configurations or combinations of these objects and actions.

To illustrate this, consider the example of the levels of information given in Chapter 3. Figure 3.8 (page 80) showed examples of system-level action patterns. Here, the source of the action patterns is exclusively the collaboration system itself: without observing the system in use, i.e. without any user input, it is possible to determine the constituent elements and structure of an action pattern such as *add_statement* shown in the figure. On the other hand, for process-level action patterns such as *Product-Concept-Development* shown in Figure 3.12 (b) (page 88), the source of the action pattern is the group of users: the specific combination of task-level action patterns that makes up the process-level action pattern is entirely dependent on the way that the collaboration system's users have combined these tasks together. Finally, Figure 3.10 (b) (page 84) shows an example of an intermediate-level action pattern, in this case the collaboration-level action pattern *Group-Discussion*. Its source is a combination of system and users: on the one hand, the basic objects and action patterns (in this case, objects *Role* and *Discussion-Forum* and action patterns *Open-Discussion-Statement* and *Post-Discussion-Statement*) are given by the collaboration system; on the other hand, their specific combination into the action pattern *Group-Discussion* is performed by the collaboration system's users.

Dependent on the source, the specification of the set of action patterns at a particular level of the Information Pyramid can be more or less *complete*, meaning that all possible action patterns are specified. Where the collaboration system is the source, the set of all possible action patterns can be feasibly determined in advance by inspecting the set of objects and actions provided by the system. Because a collaboration system provides only a finite number of actions, corresponding action patterns too are finite in number, making it possible to completely specify all possible valid action patterns at a given level of the Information Pyramid.

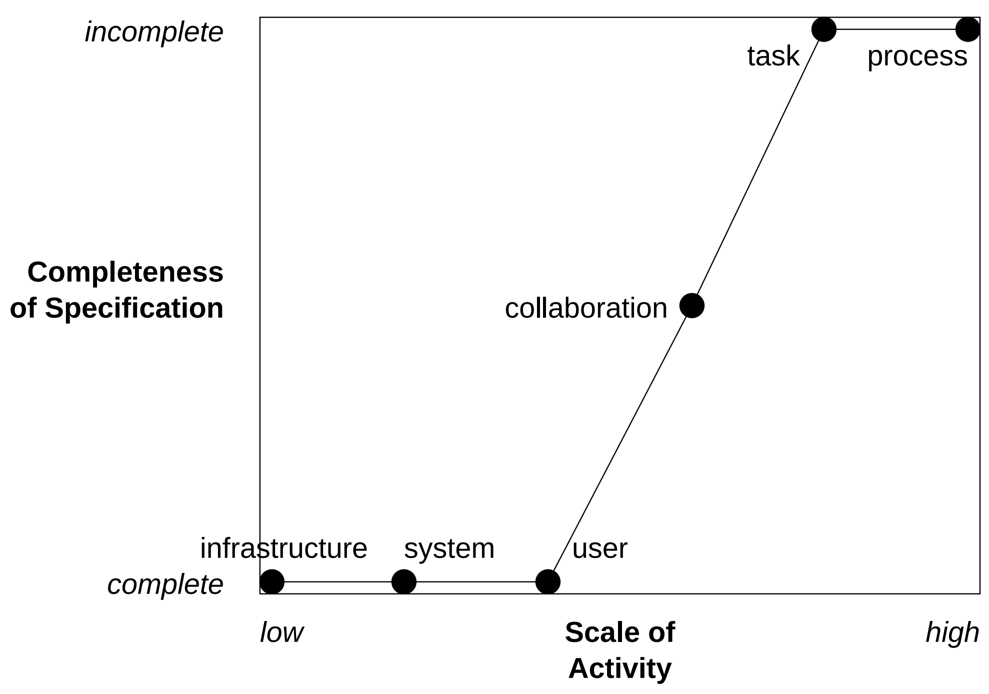


Figure 4.18: Completeness of specification versus scale of activity for the six levels of the Information Pyramid

On the other hand, where the source of action patterns is the group of users, combining objects and actions in specific ways into action patterns, the number of all possible patterns is infinite, because any number of any actions and objects can be combined in any desired way to produce a new, unique action pattern. Thus the specification of the set of action patterns at that level of the Information Pyramid is incomplete and open-ended[5]. This is illustrated in Figure 4.18, which shows how completeness of specification (on the vertical axis) is related to the scale of activity (on the horizontal axis).

In order to obtain patterns at the highest two levels, the task and process levels, therefore requires inspection of the particular combinations of objects and actions created by the system's users, and their inter-relationships. Doing so, however, is not always straightforward. A given collaboration system may contain a large number of collaboration spaces, each of which may in turn contain a large number of objects, related to one another in multiple and possibly complex ways. Thus, these patterns are often not easily *discernible* (in the sense that they can not be readily assimilated by the human observer),

[5]This is not to say that an individual action pattern may be only partially specified. Each action pattern, regardless of the level of the Information Pyramid and the source of action patterns, is always completely specified. However, the *set* of all possible action patterns at a given level of the Information Pyramid may be incomplete, in that some of the (infinitely many) possible action patterns have not been specified.

requiring time-consuming analysis of large amounts of information.

Here it is proposed that for the identification of patterns at the task and process levels of the Information Pyramid, the mentioned bottom-up approach be complemented through visual analysis and exploration of the objects and actions directly on those levels, using techniques of *information visualization*.

Information visualization aims to reveal patterns and structures in a body of information. Its advantage over the direct analysis of raw data is that it supports a more "rapid assimilation of information", thereby "reducing the time cost of information access and increasing the scale of information that a user can handle at one time" (Robertson et al., 1993). This is achieved by shifting part of the process of identifying patterns to the human perceptual system, and thus relieving the cognitive system.

Because information objects are abstract, often lacking physical representations, the challenge is to find suitable metaphors for making objects and their relationships visible (Gershon and Page, 2001). The notation used to represent information objects usually depends on the type of information that is to be visualized. Shneiderman identifies seven different basic types: 1-D, 2-D, 3-D, Temporal, Multi-Dimensional, Tree, and Network (Shneiderman, 1998, p. 524).

In the case of information about virtual collaboration, a possibly large number of objects may be related to multiple other objects. Such information can therefore be most appropriately considered to be of the network type. For instance, in the task-level patterns shown in Figure 3.12 (a) on page 88, the information items are the instances of roles, discussion forums, and documents, most of which are related to several objects of one or two other types.

For network structures, the most common forms of visualization are the *node-and-link diagram*, which represents information items as nodes connected by links representing relationships between items, and which focuses on making these relationships between items visible; and the *square matrix* where the values of a selected link attribute for pairs of items are represented in the row-column positions of a matrix and which focuses on making specific attribute values visible (Shneiderman, 1998, p. 534). Given that in the search for task and process patterns, the *structure* of the configuration of objects and actions is sought (i.e. the inter-relation and arrangement of items, cf. the discussion on structure in Section 3.1), a suitable representation is the node-and-link diagram. Such diagrams take the general form like the example shown in Figure 4.19: nodes, shown here as square boxes, represent information items; and links, shown here as connecting lines, represent relationships between these information items. Each pair of distinct nodes in the graph may be connected by one (or possibly several) link(s). This type of diagram may be used to represent an individual task, with nodes representing its constituent roles,

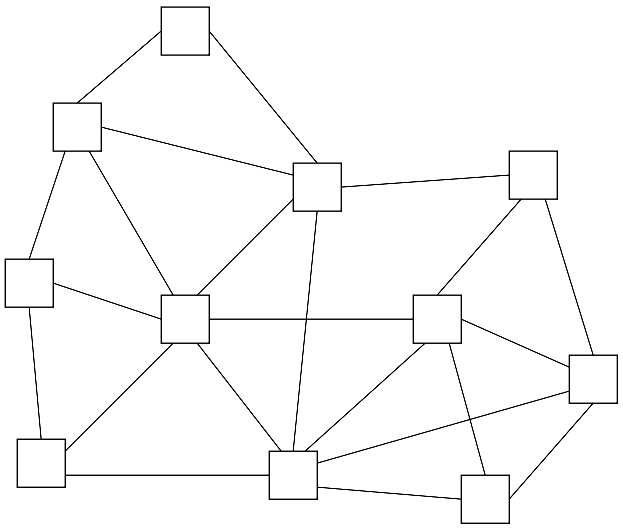


Figure 4.19: Example of a node-and-link diagram

discussion forums, documents, etc., and links representing relationships among these items. It may also be used to represent an entire process, with nodes corresponding to individual tasks and links representing inter-task relationships.

4.4.1 Measures of Collaboration Spaces

Identifying which collaboration spaces to investigate for pattern extraction itself can be non-trivial when there are many collaboration spaces. Another situation is where a number of collaboration spaces have a similar structure, and it is not immediately obvious how those collaboration spaces differ from one another. In both of these cases it can be helpful to obtain additional information about the collaboration spaces in order to facilitate their comparison. *Measures* of collaboration spaces provide such additional information.

A measure of a collaboration space is a quantitative attribute which expresses something about a certain characteristic of a collaboration space, usually an aspect of its complexity. Such a measure may be derived, or computed, from information related to the collaboration space. In order to illustrate the notion of measures of collaboration spaces, some examples of such measures are given below:

1. **Collaboration space density:** a measure of how many objects are contained in a single collaboration space. This measure can give a first indication of the complexity of the work carried out in the collaboration space, where more complex work usually involves a larger number of objects.

2. **Document exchange intensity:** a measure of how often documents are exchanged

in a collaboration space through create/read document actions per unit of time, calculated as an average over the history of the collaboration space. This measure can give an indication of the intensity of the work carried out in the collaboration space.

3. **Document exchange recency:** a measure of the number of "recent" document exchanges in a collaboration space. Recency is the number of actions (in this case document exchanges) during a fixed time interval up until the time of observation, for example the past fortnight or the past month. This measure expresses how strongly users in a collaboration space are presently exchanging documents, which can suggest the extent to which the work in the collaboration space is currently progressing.

4. **Communication intensity:** a measure of the number of statements exchanged through discussion forums in a collaboration space per unit of time, calculated as an average over the history of the collaboration space. This measure can give an indication of the intensity of the work carried out in the collaboration space.

5. **Communication recency:** a measure of the number of statements "recently" exchanged through discussion forums in a collaboration space, again for a pre-defined time interval such as a fortnight or a month. This measure expresses how strongly users in a collaboration space are presently communicating, which can suggest the extent to which work in the collaboration space is currently progressing.

6. **Evolution intensity:** a measure of how strongly the structure of a collaboration space is subject to change, in terms of change actions per unit of time, calculated as an average over the history of the collaboration space. This measure can suggest the extent to which the work carried out in the collaboration space is emergent.

7. **Evolution recency:** a measure of how strongly the structure of a collaboration space has "recently" been subject to change, again for a pre-defined time interval. This measure expresses the extent to which the work in the collaboration space is still actively evolving.

The above list of measures is by no means intended to be exhaustive but rather to be suggestive of some (largely arbitrarily selected) measures that can be useful in highlighting differences when comparing a collection of collaboration spaces. Other measures can be defined by considering other information related to collaboration spaces and expressing it quantitatively.

4.4.2 Requirements of Visualization Tools

The visualization of task and process patterns in a given collaboration system requires a visualization tool which needs to access data from that system and represent it in terms of information items and their relationships. Some functional requirements for a visualization tool are proposed here:

- **Multiple node/link types:** The ability to represent different types of information items and different relationship types differently, so as to be able to distinguish between these different types when they are contained within the same diagram.
- **Task/process visualization:** The ability to visualize networks of information at different levels of detail, corresponding to the visualization of tasks and processes.
- **Navigation:** The ability to navigate a (possibly large) network of information and focus on only a portion of the network that is of interest.
- **Filtering:** The ability to filter out information from the visible portion of a given network of information that is not of interest.
- **Comparison:** The ability to visualize measures of collaboration spaces to enable their comparison.

An example of a visualization tool that satisfies these requirements is introduced in Appendix A.

4.5 Framework for Pattern Extraction and Feedback

The preceding sections have discussed details of the modeling, specification, and derivation of patterns of virtual collaboration at different levels of the Information Pyramid. However, the ability to extract patterns of virtual collaboration from the data collected by a collaboration system is something that depends on what and how much data that system collects and makes available, which is something that can be planned and designed for. This section therefore considers the larger context of pattern extraction in the development and ongoing use of collaboration systems. It also considers how extracted patterns can be maintained and eventually fed back into ongoing use.

This involves a number of relevant topic areas: (1) collaboration systems, (2) collaboration data, (3) pattern extraction, and (4) organizational memory. Here, a *Frame-*

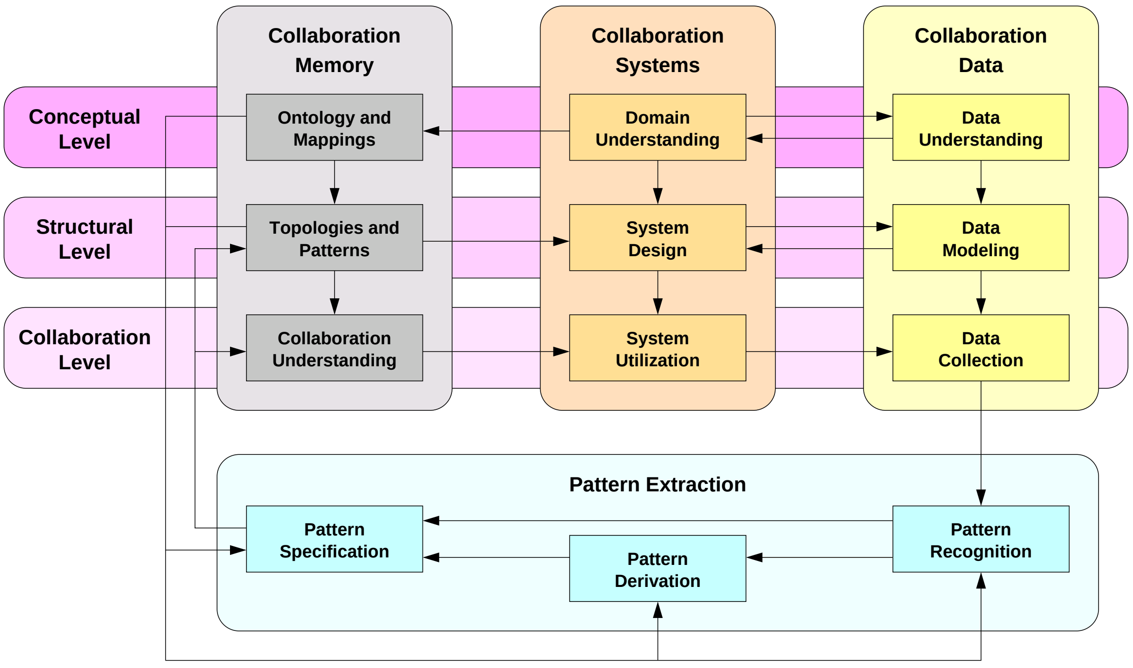


Figure 4.20: Framework integrating collaboration memory, collaboration systems, collaboration data, and pattern extraction

work for Pattern Extraction and Feedback is proposed that integrates these four different strands into a cohesive fabric[6]. The two primary goals of the framework are:

1. To influence the design of collaboration systems so as to provide the data necessary for the extraction of patterns of virtual collaboration.

2. To feed extracted patterns back into the use of collaboration systems.

Because of the pivotal role of high-quality data for pattern extraction, data design and design of the collaboration system are seen as complementary and parallel activities, affording the opportunity to more greatly control the extent and quality of data collection. Moreover, patterns extracted from collaboration data can themselves contribute to the ongoing design of a collaboration system. A number of related research efforts are underway in the direction of controlled data collection, carried out mainly in the field of e-commerce and Web data mining (Ansari et al., 2000; Spiliopoulou and Pohle, 2001).

A graphical depiction of the framework is shown in Figure 4.20. It includes four major groups of inter-woven components:

[6]This framework is based on earlier joint work of the present author and Dr. Simeon J. Simoff (Biuk-Aghai and Simoff, 2001).

1. **Collaboration Systems:** concerned with the analysis, design, implementation, and utilization of the core collaboration support technology.

2. **Collaboration Data:** concerned with the analysis, design, and collection of data related to collaboration.

3. **Pattern Extraction:** concerned with the analysis of data, recognition and extraction of patterns, and derivation and specification of progressively higher-level patterns.

4. **Collaboration Memory:** concerned with maintaining concepts and making available patterns of collaboration at various levels of abstraction.

Moreover, the three components appearing in the upper part of the figure consist of three parts, at different levels of abstraction:

1. **Conceptual Level:** related to elementary concepts.

2. **Structural Level:** related to designs and structures intended for use.

3. **Collaboration Level:** related to actual collaboration instances.

Between the different parts, shown as rectangular boxes in the figure, arrows indicate flows of data and/or information. Below, each of the components of the framework is discussed in more detail, starting from the centre (*Collaboration Systems*), then continuing clockwise through *Collaboration Data* and *Pattern Extraction* to *Collaboration Memory*.

4.5.1 Collaboration Systems

The first major component of the framework is related to collaboration systems, the support systems through which collaboration is performed. When a new collaboration system is designed and developed, certain decisions are made as to the basic features of the collaborative work domain it should support. Such basic features may include, for instance, synchronous vs. asynchronous work, formal vs. informal collaboration, loosely vs. tightly coupled work, etc. Designing and developing a collaboration system, therefore, should entail obtaining an understanding of such domain-dependent requirements for the system.

The activities from conception to implementation and use of the collaboration system are shown in the vertical dimension of the box labeled *Collaboration Systems* in Figure 4.20, proceeding top-down from the abstract (*Domain Understanding*) through the intermediate-level (*System Design*) to the concrete (*System Utilization*):

4.5.1.1 Domain Understanding

Domain understanding (the top box in the figure) refers to the study of the domain of collaboration to be supported by the new collaboration system, i.e. the basic features of collaborative work touched upon above. In terms of software development, this corresponds to the step of requirements analysis. Decisions made at this stage determine what kind of collaboration system will result from the development process. Some fundamental features to be supported by the system may also be laid down at this stage as requirements, including: the structuring metaphor to be employed, navigation facilities, representation of people and their abilities, provision of awareness information, access to various information sources and tools, etc. When completed, this step contributes to an understanding of the domain of collaborative work, its concepts, activities, objects involved, etc. Together with feedback from the step of *Data Understanding* (discussed later), this understanding of the domain is specified in the form of a multi-level ontology corresponding to the Information Pyramid, as well as mappings between levels.

4.5.1.2 System Design

The conceptual modeling phase is followed by the design phase (the centre box in the figure) where the identified requirements are translated into software designs. At this phase, decisions are made about the details of the implementation of each conceptual element, and how each requirement is to be satisfied in the new system. If not already fixed during the previous phase, decisions are made on the type of interface and interaction mode (such as textual vs. graphical, web-based vs. client-server), representations of collaboration spaces and of all conceptual elements in the collaboration space, abilities and affordances these conceptual elements are to be furnished with and the relationships they can have with one another, layout and detailed structuring of the collaboration space, navigation paths, etc. After the design has been finalized, it is implemented as a working collaboration system.

System design, however, extends even beyond software design and implementation. Collaboration systems, including all those surveyed in Chapter 2, offer their users the ability to configure and customize their collaboration spaces. This may be as simple a matter as adding a few documents and users into a collaboration space, or as complex as designing entire processes consisting of multiple tasks spread across several collaboration spaces and involving multiple users, communication channels, and artefacts each. Thus even in the finished collaboration system, a certain amount of "system design" is carried on continuously. This second type of system design creates the structures within which actual collaboration takes place. Different collaboration requirements, as well as working styles and personal preferences, and not least of all familiarity with the system, influence the way these collaboration spaces are set up, i.e. what virtual structures users create. While the collaboration system itself provides users with virtual "spaces", it is through the appropriation of these spaces, their configuration, and by populating them with the necessary props that they become "places" for collaboration, to use the terminology of (Harrison and Dourish, 1996).

4.5.1.3 System Utilization

Once a collaboration system has been created, and collaboration spaces have been set up and configured, these are utilized for collaboration (the lower box in the figure). At this time, users enter the collaboration spaces and carry out the activities for which these were designed. During this phase, it is possible that changes may be made to collaboration spaces in response to changes in the processes carried out in them. That is, in highly emergent processes, the activity of design of collaboration spaces may be ongoing into their utilization, and the two activities may be interleaved with one another throughout the collaboration space's existence. In other cases, where a collaboration process is more stable and predictable, the collaboration space may be utilized with no or little change throughout its existence.

4.5.2 Collaboration Data

The second major component of the framework, shown in the right hand portion of Figure 4.20, is related to collaboration data. Within the framework, collaboration data is understood to be that portion of data related to concepts within the domain of the collaboration system. This includes data on the objects and actions provided by the collaboration system. Some of this data is of direct use within the collaboration system, while other data exists solely for the purpose of facilitating later pattern extraction.

Traditionally, the majority of collaboration systems have only maintained a small portion of what is here referred to as collaboration data, namely "internal data" needed for their own operation. However, for purposes of pattern extraction, this portion of data is almost always insufficient—either by lacking data on some of the concepts of interest, by lacking sufficient detail, or by lacking enough contextual information to enable different data items to be related during pattern extraction.

To remedy this deficiency and to emphasize the importance of collaboration data, the status of design and collection of collaboration data is elevated by treating it as a separate component within the framework. The activities related to collaboration data proceed, as shown in the vertical dimension of the right hand portion of the figure, from the abstract

(*Data Understanding*) through the intermediate-level (*Data Modeling*) to the concrete (*Data Collection*):

4.5.2.1 Data Understanding

Data understanding is the first activity in the sphere of collaboration data. Its purpose is to obtain an understanding of the main data elements required in a collaboration system. Data understanding thus essentially consists of conceptual data modeling. However, the framework does not impose a specific modeling method, so the outcome of this activity may be an E-R model, an object model, or whatever output the modeling method produces.

The activity of data understanding goes hand-in-hand with the corresponding activity on the conceptual level of collaboration systems, i.e. with *Domain Understanding*. While domain understanding leads to the identification of concepts in the domain under investigation, data understanding leads to the identification of data elements needed to represent these concepts. This relationship is represented by the information flow between *Domain Understanding* and *Data Understanding*. On the other hand, the activity of data understanding may also advance domain understanding, for instance by identifying details of concepts or relationships previously not considered, perhaps even suggesting the inclusion of additional concepts. This is represented by the reverse flow, from *Data Understanding* to *Domain Understanding*.

4.5.2.2 Data Modeling

Once data understanding is complete, the initial data requirements are transferred to the next stage, *Data Modeling*. More specifically, this activity is equivalent to the step of logical, and subsequently physical, data modeling. As before, this activity too goes hand-in-hand with the corresponding activity of collaboration systems on the same level, i.e. with the structural-level activity *System Design*. While in the collaboration systems' sphere decisions relating to every aspect of the collaboration system are made, the parallel activity of data modeling makes decisions on details of collaboration data, their relationships, and representation. This includes such aspects as the data model employed (relational, object-oriented, hierarchical, etc.); details of all data elements, including fields, data type and size; relationships among data elements; storage organization and file formats; etc.

4.5.2.3 Data Collection

Finally, once the data model has been completed, and subsequently implemented, the collaboration system is put to use and data is collected. Once again, the activity at this level, *Data Collection*, works in parallel with the corresponding collaboration systems activity, *System Utilization*, with data transfer from the latter to the former. That is, utilization of the collaboration system generates data which is transferred for data collection, shown by the flow from *System Utilization* to *Data Collection*.

Minimally, data collection simply involves storing data in a certain form, such as in a database or in flat files. However, it may also involve some simple processing of the data. Such processing may be performed in preparation for subsequent pattern extraction. This may involve sorting, grouping, or other operations. These processing steps are driven by the decisions made during the data modeling phase, indicated by the arrow between *Data Modeling* and *Data Collection* in Figure 4.20. The aim of data collection is thus not only to make data available, but ideally to make data available in a form which is ready for pattern extraction without the need for additional pre-processing.

4.5.3 Pattern Extraction

The third major component of the framework is related to pattern extraction, shown as the horizontal box at the bottom of Figure 4.20. Pattern extraction is concerned both with identifying new patterns of virtual collaboration, as well as with obtaining instances of those patterns from collaboration data, using the methods described earlier in this chapter. It consists of three activities, shown right to left in the box labeled *Pattern Extraction*, proceeding from *Pattern Recognition*, through *Pattern Derivation*, to *Pattern Specification*:

4.5.3.1 Pattern Recognition

The first activity related to pattern extraction is the recognition of patterns in the source body of data (shown as the right-most box in the figure). The data that forms the input to this activity is obtained from the collaboration system, which collects it (in activity *Data Collection*) in a form as previously designed during the activity of *Data Modeling*. This source data is analysed to first identify instances of elementary concepts, i.e. objects and actions at the base level of the Information Pyramid. Once instances of these objects and actions have been recognized in the data, instances of action patterns involving these objects and actions are extracted. The recognition of instances of objects and actions, and the extraction of action patterns, draws upon the corresponding classes previously identified and specified during the activity of *Domain Understanding* and provided by *Ontology and Mappings* (discussed later).

4.5.3.2 Pattern Derivation

Once instances of patterns at the base level of the Information Pyramid have been recognized, these provide the input for deriving instances of higher-level patterns (represented by the arrow from *Pattern Recognition* to *Pattern Derivation*). That is, from instances of base-level action patterns identified during *Pattern Recognition*, instances of action patterns on the next-higher level may be derived. These instances of derived patterns may then in turn be used to identify instances of patterns on the next-higher level, and so on up to the top level of the Information Pyramid.

Besides deriving instances of patterns, this activity may also identify new classes of action patterns not previously observed. While action patterns on the micro and meso levels of the Information Pyramid can be completely specified, it is not possible to completely specify all possible action patterns on the macro levels of the Information Pyramid, as these are infinite in number (cf. the discussion in Section 4.4). Particularly in the case of collaborative processes that are emergent, new configurations of collaboration spaces, and networks of related collaboration spaces, are created by the collaboration system's users. These may be identified by drawing upon and analysing instances of lower-level objects, actions, and action patterns. As discussed earlier, in Section 4.4, this activity can be supported through information visualization. The outcome are new classes of action patterns on the macro levels of the Information Pyramid which contribute to an ever-growing ontology of the given collaboration system.

Both the derivation of instances of action patterns, as well as the identification of new classes of action patterns draws upon definitions of classes of the concepts involved, as well as definitions of mapping functions, both of which, as above in the case of the *Pattern Recognition* activity, are provided by *Ontology and Mappings* (discussed later).

4.5.3.3 Pattern Specification

The last step of pattern extraction is the representation of the discovered patterns in a form corresponding to the specification of the related classes in the ontology. This consists of the specification of both discovered instances of patterns, as well as newly derived classes of patterns. The input to this activity is produced by the activities *Pattern Recognition* (for instances of patterns) and *Pattern Derivation* (for both instances and new classes of patterns). As with these earlier two activities, pattern specification draws upon the existing specification of elementary classes (namely of objects and actions, which are the basic building blocks of action patterns), as well as that of other action pattern classes.

Extracted patterns are deposited in *Collaboration Memory*: newly discovered pattern classes feed into *Topologies and Patterns*, while discovered instances of patterns feed into *Collaboration Understanding* (this is discussed in more detail below).

4.5.4 Collaboration Memory

The final major component of the framework is the box labeled *Collaboration Memory*, shown in the left hand portion of Figure 4.20.

Chapter 2 discussed organizational memory as one of the core areas which constitute the problem domain of this research. There, the distinction between declarative and procedural memory was made, and the need for capturing more of the procedural aspect of organizational work processes was pointed out (cf. pp. 59–60). Applied to virtual collaboration, this suggests that there is value in retaining and later drawing on historical records of such collaboration. Such records may be referenced when setting out on new virtual collaboration, to “see how others have done it”, and perhaps to reuse and re-enact parts of others’ experience. They also help provide awareness on what has been “going on” during collaboration. In this way, they have the potential to help address the challenges posed in Chapter 1 (cf. pp. 6–8).

Here, the notion of a *collaboration memory* is proposed, building upon the suggestion of Conklin to link groupware with organizational memory (Conklin, 1993). Collaboration memory is defined as follows:

> **Definition 19** *A **collaboration memory** constitutes one part of an organizational memory, consisting of records of procedural aspects of collaborative activity.*
> □

That is, an organizational memory is understood to consist of multiple parts (cf. also (Ackerman and Halverson, 2000)), of which collaboration memory is but one. These different parts of organizational memory each capture different aspects of an organization’s activity and outcomes, and should be regarded as complementary. That is, while some parts of the overall organizational memory may contain declarative descriptions of, say, a problem and the resulting solution that was devised for it, the collaboration memory complements this with a procedural description of the collaboration that brought about the solution. In the case of virtual collaboration, these procedural descriptions are the patterns of virtual collaboration discussed in this and the preceding chapter.

Each of the parts of collaboration memory in this framework is discussed below, proceeding top-down from *Ontology and Mappings*, through *Topologies and Patterns* to *Collaboration Understanding*:

4.5.4.1 Ontology and Mappings

The first part of the collaboration memory is related to maintaining the ontology of elementary concepts on the different levels of the Information Pyramid, and mappings between these levels. This consists of specifications of objects and actions, which form the basic “building blocks” for more complex entities such as action patterns. These objects and actions are specified using information on these basic concepts that is obtained from the activity of *Domain Understanding* (this is indicated by the arrow from that activity to *Ontology and Mappings*).

The ontology of these elementary concepts and mappings, as well as patterns (discussed below), feeds into each activity belonging to *Pattern Extraction*. That is, pattern recognition, derivation, and specification rely on these concepts and mappings in order to recognize instances of these concepts in the source data, to map lower-level concepts to higher-level concepts, and to finally specify extracted patterns in a form according to the ontology’s definition of the concepts involved. This is indicated by the arrows from *Ontology and Mappings* to each of the three boxes in *Pattern Extraction*.

4.5.4.2 Topologies and Patterns

The second part of the collaboration memory is related to maintaining definitions of classes of patterns of virtual collaboration. These are expressed in terms of the elementary concepts of objects and actions deposited in the *Ontology and Mappings* part of the collaboration memory (this is indicated by the arrow from that part to *Topologies and Patterns*). Thus, as seen in the discussion of action patterns in Chapter 3, a given action pattern class usually contains references to multiple object and action classes. In this part of the collaboration memory, only *classes* of patterns are deposited, not actual instances of these classes.

Patterns deposited at this level can be of use in the design of new collaboration spaces. That is, when a new collaborative task or process is being embarked on, existing patterns of the same or similar tasks or processes can be referenced to aid in the setup of the new collaboration space(s) (this is indicated by the arrow from *Topologies and Patterns* to *System Design*). These retained patterns thus constitute certain reusable *topologies* of the structure of collaboration spaces.

Together with elementary concepts of objects and actions from the part of *Ontology and Mappings*, the patterns in this part of the collaboration memory feed into the activities of *Pattern Extraction*—as was discussed above. Again, this is indicated by the arrows from *Topologies and Patterns* to each of the three boxes in *Pattern Extraction*.

On the other hand, during *Pattern Extraction* new classes of patterns may be discovered. Once these are specified in the appropriate form, referencing the elementary concepts of the objects and actions involved, these feed back into the collaboration memory in the part of *Topologies and Patterns*, thus becoming available for later use.

4.5.4.3 Collaboration Understanding

The third part of the collaboration memory is related to maintaining instances of patterns of virtual collaboration. These are expressed as instances of the relevant pattern classes that are deposited in the *Topologies and Patterns* part of the collaboration memory (this is indicated by the arrow from that part to *Collaboration Understanding*). These instances express how collaboration is actually being performed, by capturing actual values for the attributes of the collaboration that make up the corresponding pattern classes. This may also include *derived attributes* that characterize features of the collaboration, such as the attributes measuring properties of group discussion shown in the example in Table 3.3 on page 83.

The source of these instances of patterns of virtual collaboration is the block of activities of *Pattern Extraction*. There, these instances of patterns are extracted from the source data and specified in the form of instances of the corresponding classes of patterns. This is indicated by the arrow from *Pattern Specification* to *Collaboration Understanding*.

The instances of patterns contained in this part of collaboration memory can provide an understanding of the collaboration to those requiring it—members of the virtual teams themselves, or other stakeholders such as management. For example, it can identify what main types of activities were conducted within a collaboration space, how the activities were carried out over time, what differences exist in the activity of different people within the collaboration space, etc. Thus they have the potential to serve as an awareness resource within the collaboration spaces from which they originated. This is indicated by the arrow from *Collaboration Understanding* to *System Utilization*.

4.6 Summary

This chapter has elaborated on the extraction and derivation of patterns of virtual collaboration from the data collected from a collaboration system.

It was proposed that this process of pattern extraction and the derivation of higher-level patterns requires an ontology of concepts and mappings related to each level of the Information Pyramid to be specified. A modeling method for creating such an ontology step-by-step, starting from the lowest level and working its way up to the top level, was presented. In order to actually specify the ontology, both a textual and a graphical notation were introduced.

Details of several issues related to the specification of the ontology were given. Regardless of the actual collaboration system for which an ontology is to be created, a set of common classes can be defined, thus forming a *base* for the new ontology. When identifying patterns of virtual collaboration according to the specifications contained within the ontology, it may be necessary to distinguish between different *sessions* within which the patterns' constituent actions take place. For the mapping of the ontology's concepts across levels of the Information Pyramid, principles and mechanisms were developed which distinguish between three different kinds of correspondences of concepts. Once correspondences have been identified, *mapping functions* are specified which transform concepts across levels.

The degree of completeness that can be achieved in the specification of patterns depends on the level of the Information Pyramid being specified, decreasing from bottom to top level. This is paralleled by a shift of the source of patterns from the collaboration system to its users. To facilitate the recognition of patterns on the highest levels of the Information Pyramid, it is proposed to employ techniques of *information visualization*.

Finally, a Framework for Pattern Extraction and Feedback was proposed which places the activity of pattern extraction and derivation in the larger context of the development and utilization of collaboration systems. Moreover, the framework suggests how discovered patterns can be retained in a special type of organizational memory termed *collaboration memory*, as well as how these retained patterns can feed back into the design and use of collaboration spaces.

Having detailed the modeling and transformation of information about virtual collaboration in this and the preceding chapter, the following chapter goes on to apply the proposed concepts and methods to the extraction of patterns of virtual collaboration from an actual collaboration system.

Chapter 5

Case Study: Modeling and Pattern Extraction in LIVENET

The previous two chapters have detailed the concepts and methods involved in modeling and extracting patterns of virtual collaboration. It was shown that this involves on the one hand the specification of an ontology of concepts in the domain of collaboration and for the particular collaboration system that constitutes the source of data; and on the other hand the specification of mappings for correspondences between concepts on different levels of abstraction. Given these two, the models and the mappings, it is possible to transform source data that represents detailed, small-scale activity through successive mappings into abstract representations of large-scale collaborative activity, in the form of patterns of virtual collaboration.

This chapter demonstrates the use of the concepts and methods by means of a case study, providing some degree of validation of their plausibility and applicability. The given case study involves student users engaged in the collaborative preparation of reports, using a particular collaboration system, LIVENET, and the data collected by it (LIVENET was first introduced in Chapter 2, pp. 45–48). Since its initial development, this system has evolved over several versions; the material in this chapter is based on LIVENET version 2.4. LIVENET has been used over the course of several semesters to support courses involving hundreds of users, including students and faculty at the University of Technology, Sydney as well as other institutions. The testing and on-going development of LIVENET that has led up to version 2.4 have produced a well-functioning system that is a reliable source of data for use in this case study.

As presented in the previous two chapters, both the concepts and methods proposed are generic and are not limited to any specific collaboration system, type of users, or activity. As is the case in any case study, the application of concepts and methods here is naturally specific to the particular collaboration system, users, and activities that are the subject of this case study. However, this specific application of concepts and methods in the given case study should not be regarded as implying any limitation in their applicability to cases similar to the one presented here. On the contrary, since the concepts and methods developed in this thesis are generic by design, they can be equally applied in other collaboration systems, with other types of users, as well as with other types of activities.

The following two sections concentrate, respectively, on the specification of the LIVENET ontology, and on the application of the ontology to actual data collected from the LIVENET system.

5.1 Specification of the LIVENET Ontology

The first step in the process leading to the extraction of patterns of virtual collaboration consists of modeling the concepts of the Information Pyramid for the collaboration system that is the source of data, in this case the LIVENET system. This produces an ontology consisting of specifications of classes of concepts, as well as mappings between corresponding classes across levels. The method followed is the one described in Chapter 4, and briefly outlined below:

1. Identify the base level of the Information Pyramid for the given system.
2. Model the base level.
3. Model the next-higher level.
4. Define mappings between the two levels just modeled.
5. Repeat the previous two steps until the top level of the Information Pyramid is reached.

For the modeling of the LIVENET system, each of these steps is carried out below. However, as a complete specification of the ontology for LIVENET would fill many pages and would go beyond the scope of this thesis, specification is mainly illustrated in the first few steps below, while detailed specification is omitted in later steps. The specifications of concepts that are shown lead from detailed ones on lower levels up to a final action pattern of manuscript preparation at the process level.

5.1.1 Step 1: Identifying the Base Level

In the case of the LIVENET collaboration system, a number of information sources exist (the level of each of these is shown in parentheses):

- **Web server log** (infrastructure level): the LIVENET system is web-based, and so records of user interactions are maintained in the web server log. These records contain mostly dynamic information, and only to some extent static information. However, both dynamic and static information are expressed in terms of web page accesses, and as the requested pages are themselves dynamically produced, it is difficult to relate this information to system-level entities.

- **Database transaction log** (infrastructure level): the LIVENET system's application data is stored in a database management system (DBMS), and so the transaction log of the DBMS constitutes a record of dynamic information. This log contains great detail, however its focus are the operations accessing or manipulating data contained in the LIVENET database, rather than the data itself.

- **Database tables** (system level): LIVENET application data is stored in a relational database. Information on different kinds of entities, such as users, documents, workspaces, etc., are stored in separate database tables. This information reflects the current state of each LIVENET entity, but not its history.

- **LIVENET server log** (system level): LIVENET is implemented as a client-server system, with the server receiving requests (commands) from the client, which it in turn carries out. These client requests are recorded on the server side in the form of a server log. It contains details about the types of requests, as well as references to relevant LIVENET entities such as workspaces, users, etc. The server log thus constitutes a detailed source of system events.

Given that both static and dynamic information are available on the system level (contained in application data and system events, respectively), it is not necessary to make use of the infrastructure level data sources. Therefore the base level of the Information Pyramid for LIVENET is the *system level*.

5.1.2 Step 2: Modeling the Base Level (System Level)

Given that the system level has been identified as the base level, its concepts can now be modeled. As described in Section 4.1.2 (pp. 94–95), this consists of the steps of identifying objects, actions, and action patterns, and specifying these concepts.

5.1.2.1 Step 2.1: Identifying Objects

On the system level, information related to objects in LIVENET is deposited in a relational database consisting of 16 tables. These contain information on following 18 types of objects, most of which are stored in separate database tables:

Object Name	**Description**
Action	a tool for performing some action, callable from within a workspace
Background	similar to a document, but containing information that is intended as “background material” for a given task
Block	a collection of discussion statements
Discussion	a reference to a discussion forum, as visible to the user
Document	an artefact containing information, residing on a network server
Forum	an internal representation of a discussion forum
Message	a semi-structured message routed between workspaces according to a pre-defined message rule
Message-Rule	a rule defining routing of messages between workspaces, based on their message type
Message-Type	a defined type of a message in a workspace
Notification-Rule	a rule specifying which user to notify by external email when a statement is posted in a discussion forum
Object	a thing of a particular type residing in a workspace; Document, Background, Action, and Discussion are types of objects
Participant	a user occupying a particular role in a particular workspace
Role	an organizational role occupied by users in a workspace
Statement	a statement posted in a discussion forum
User	a person registered in the system
User-Group	a collection of users
Workgroup	a conceptual entity grouping together users and workspaces
Workspace	a collaboration space

Each of these entities needs to be modeled as an object concept in the system-level part of the ontology.

5.1.2.2 Step 2.2: Identifying Actions

System-level dynamic information is contained in a server log file consisting of log records. These capture information on commands performed (i.e. actions), and contain references to system-level objects involved. In the case of LIVENET, technical documentation of all server commands exists, making the identification of the set of system-level actions trivial[1]. In total, LIVENET has 70 different commands, thus corresponding to 70 actions, listed in brief below:

Action Name	**Description**
Add-Block	Add a new block to a forum
Add-Forum	Add a new forum
Add-Group-User	Add a user to a group
Add-Message	Add a message
Add-Msg-Rule	Add a new message rule to the current workspace
Add-Msg-Type	Add a new message type to the current workspace
Add-Notify	Add a notification email address
Add-Object	Add an object to the current workspace
Addparticipant	Add a new participant to the current workspace
Add-Role-Object	Assign an object to a role
Add-Statement	Add a new statement to a block
Add-User	Create a new user
Change-Path	Change the path of an object
Change-Type	Change the type of an object
Create-Workgroup	Create a workgroup
Create-Workspace	Create a workspace
Delete-Block	Delete a block from a forum
Delete-Forum	Delete a forum
Delete-Group-User	Delete a user from a group
Delete-Message	Delete a message
Delete-Msg-Rule	Delete a message rule from the current workspace
Delete-Msg-Type	Delete a message type from the current workspace
Delete-Notify	Delete a notification email address
Delete-Object	Delete an object from the current workspace
Deleteparticipant	Delete a participant from the current workspace

continued...

[1]For other collaboration systems which do not have such documentation, the set of commands has to be discovered through examination of the server log.

Action Name	**Description**
Deleterole	Delete a role from the current workspace
Delete-Statement	Delete a statement
Delete-User	Delete a user
Delete-Workgroup	Delete a workgroup and all workspaces in the workgroup
Delete-Workspace	Delete a workspace
Edit-Statement	Edit an existing statement
Edit-User	Edit an existing user
Exe-External	Launch an external program
Get-All-Objects	Get all the objects of the current workspace
Get-All-Workgroups	Get all the workgroups in the system
Get-Block-Tree	Get the statement tree of a block
Get-Led-Workgroups	Get workgroups of which the current user is a leader
Getlogin	Get login information of the specified user
Get-Msg-Rules	Get the message rules of the current workspace
Get-Msg-Types	Get the message types of the specified workspace
Getmyworkspaces	Get workspaces of which the current user is a participant
Get-Namevalues	Get server configuration parameters
Get-Own-Workspaces	Get workspaces of which the current user is the owner
Getparticipants	Get all participants of the current workspace
Get-Role-Messages	Get all the messages the current user can access
Get-Role-Objects	Get the objects that a specific role can access
Getroles	Get roles of a given workspace
Get-Role-Templates	Get roles and their permission templates of a given workspace
Get-Statement	Get a statement
Getstatistics	Get server statistics
Get-User-Email-Homepages	Get all users' email addresses and homepage URLs
Get-User-Emails	Get all users' email addresses
Get-User-Names	Get all users' names
Get-User-Profiles	Get all users' profiles
Get-Users	Get all users' information
Get-Users-In-Group	Get all the users in a given group
	continued...

Action Name	**Description**
Get-Users-In-Mygroups	Get all the users in the groups of which the current user is a leader
Get-Workspace-Tree	Get workspace tree
Get-Ws-Objects	Get objects owned by the current user
Give-Ownership	Change the owner of the current workspace to another user
Login	Log a user in
Logoff	Log a user off
Modify-Workspace	Modify an existing workspace
Newrole	Create a new role in the current workspace
Open-Object	Open an object in the current workspace
Remove-Role-Object	Remove an object from a role
Send-Email	Send email
Set-Role-Template	Set a role's permission template
Set-Role-Url	Set a role's URL
Setworkspace	Enter a workspace

Most of these actions take one or more action attributes (some take as many as ten). Action attributes may provide information needed for the execution of the action, and may also identify objects in the LIVENET database that are involved in the action.

5.1.2.3 Step 2.3: Identifying Action Patterns

To identify action patterns involving the actions identified above, the specific action context involved in each action has to be identified too. This can be discovered by inspecting the source data and determining the action context associated with each action.

According to Definition 16 (p. 69), "an action context is the set of information identifying the subject, referent, location, and time of an action." In the case of the commands recorded in the LIVENET server log, information on context is included in each log record. An example of three log records was shown earlier, in Section 3.4.1 (p. 78), and the structure of each log record was shown in Table 3.1 (p. 79). The four parts of the action context provided in LIVENET's server log records are:

Subject: identification of the action performer

- user performing the action
- role of the user performing the action

Referent: that which is being acted upon

- reference to the object being affected by the action

Location: identification of the logical/virtual location of the action

- workgroup in which the action occurred
- workspace in which the action occurred
- session in which the action occurred

Time: moment in time of the action's performance

- timestamp of the action

The detailed content of each action context depends on the action which it belongs to. For instance, the Add-Statement action, which adds a discussion statement to an object of type Block, takes a reference to the Block to which the statement is being added as its referent (all other parts of the action context are as described above). On the other hand, the Login action, which logs a user into the LIVENET system, has no referent (no object is being affected by the login action), no subject (because at the time the action is performed, no reference to the performing user exists yet), and besides a session identifier no location information.

Thus in order to identify action patterns for LIVENET's system-level actions, the action context has to be identified for each action individually. However, all actions have *structurally* only one possible context. That is to say, if a given type of action includes, for example, a role as its subject and a discussion forum as its referent, that same type of action will always include those two types of objects in those parts of the action context, and cannot be used, say, without a discussion forum or with two roles.

Thus the action patterns can be identified in a straightforward manner by identifying the object types involved in the action context of each action type. Consequently there are 70 action patterns corresponding to the 70 actions identified earlier.

5.1.2.4 Step 2.4: Specifying Concepts

The object, action, and action pattern concepts identified above now need to be specified. This specification can be based on the common classes defined in Section 4.2.1 (p. 101). Thus the classes can be created as subclasses of the common classes `Sys-Lvl-Object`, `Sys-Lvl-Action`, and `Sys-Lvl-Action-Pattern`. Below, this is illustrated for the object Statement and action Add-Statement, as well as the action pattern Add-Statement-Pattern involving both this object and this action.

The first class, `LN-Statement`, models the object concept Statement, a statement in a LIVENET discussion forum:

```
(defclass LN-Statement "A statement posted in a discussion forum."
   (is-a Sys-Lvl-Object)
   (role concrete)
   (single-slot Block
      (type INSTANCE)
      (allowed-classes LN-Block)
      (cardinality 1 1))
   (single-slot StatementNo
      (type INTEGER)
      (cardinality 1 1))
   (single-slot Type
      (type STRING)
      (cardinality 0 1))
   (single-slot Originator
      (type INSTANCE)
      (allowed-classes LN-User)
      (cardinality 1 1))
   (single-slot ParentStmt
      (type INSTANCE)
      (allowed-classes LN-Statement)
      (cardinality 0 1))
   (single-slot Heading
      (type STRING)
      (cardinality 1 1))
   (single-slot Text
      (type STRING)
      (cardinality 0 1))
   (single-slot ArtefactLink
      (type STRING)
      (cardinality 0 1))
   (single-slot DateTimeSent
      (type INTEGER)
      (cardinality 1 1)))
```

The class name has the prefix `LN-` to indicate that it models a LIVENET concept.

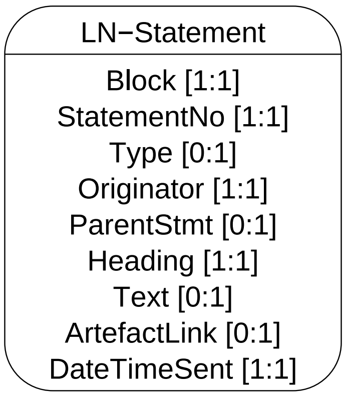


Figure 5.1: System-level object class `LN-Statement`

Using such a naming convention effectively establishes a separate namespace for each collaboration system modeled. The class subclasses the `Sys-Lvl-Object` class, thereby establishing that it represents a system-level object type. It has a number of slots which can be filled with values; in this case, these are all single-slots, meaning that they can contain only a single value. Some of these slots are intended to contain instances of other classes, such as the `Originator` slot whose value is constrained to an instance of the `LN-User` class (a class representing LIVENET users). Moreover, each slot specifies the allowed cardinality, in this case being either $0\ldots1$, meaning that a value for this slot is optional, or $1\ldots1$, meaning that a value for this slot is required. A graphical representation of the class is shown in Figure 5.1.

Next, the class `LN-Add-Statement` is specified, modeling the action concept Add-Statement which adds a discussion statement to a discussion forum:

```
(defclass LN-Add-Statement "An action that adds a statement to a
                            discussion forum."
   (is-a Sys-Lvl-Action)
   (role concrete)
   (single-slot Heading
      (type STRING)
      (cardinality 1 1))
   (single-slot Text
      (type STRING)
      (cardinality 0 1)))
```

This action is very simple, having only two attributes, the statement heading and text which are being added to the discussion forum. In graphical form, this action class is

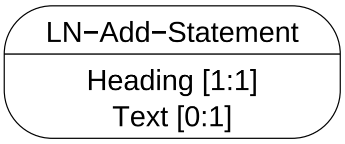


Figure 5.2: System-level action class `LN-Add-Statement`

shown in Figure 5.2.

When this action is performed in LIVENET, it is situated within a context that includes a subject consisting of the user and role carrying out the action; a referent that identifies the block to which the new statement is to be added; a location identifying the workgroup and workspace, as well as the session, in which the action occurs; and a timestamp for the moment when the action is performed.

When modeling the action pattern corresponding to this action, both the action itself, as well as all elements of its context need to referenced. Below is the specification of class `LN-Add-Statement-Pattern` corresponding to action pattern concept Add-Statement-Pattern:

```
(defclass LN-Add-Statement-Pattern "An action pattern that adds a
                                    statement to a discussion forum."
   (is-a Sys-Lvl-Action-Pattern)
   (role concrete)
   (single-slot Action-Instance
      (type INSTANCE)
      (allowed-classes LN-Add-Statement)
      (cardinality 1 1))
   (single-slot User
      (type INSTANCE)
      (allowed-classes LN-User)
      (cardinality 1 1))
   (single-slot Role
      (type INSTANCE)
      (allowed-classes LN-Role)
      (cardinality 1 1))
   (single-slot Block
      (type INSTANCE)
      (allowed-classes LN-Block)
      (cardinality 1 1))
```

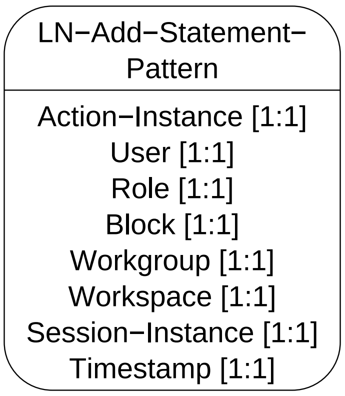


Figure 5.3: System-level action pattern class `LN-Add-Statement-Pattern`

```
   (single-slot Workgroup
      (type INSTANCE)
      (allowed-classes LN-Workgroup)
      (cardinality 1 1))
   (single-slot Workspace
      (type INSTANCE)
      (allowed-classes LN-Workspace)
      (cardinality 1 1))
   (single-slot Session-Instance
      (type INSTANCE)
      (allowed-classes Session)
      (cardinality 1 1))
   (single-slot Timestamp
      (type INTEGER)
      (cardinality 1 1)))
```

Here the class's first slot references an instance of the action of type `LN-Add-Statement`, while all remaining slots constitute the action's context. This class is shown in graphical form in Figure 5.3.

Specified classes in the ontology usually are related to other classes. This was seen in the specification of the `LN-Add-Statement-Pattern` class which references instances of seven other classes. Represented graphically, the relationships between the action pattern and its action on the one hand, and all involved objects on the other hand become visible. This is shown in Figure 5.4 which shows the action pattern `LN-Add-Statement-Pattern` together with action `LN-Add-Statement` and six related objects (arrows symbolize the *references* relationship).

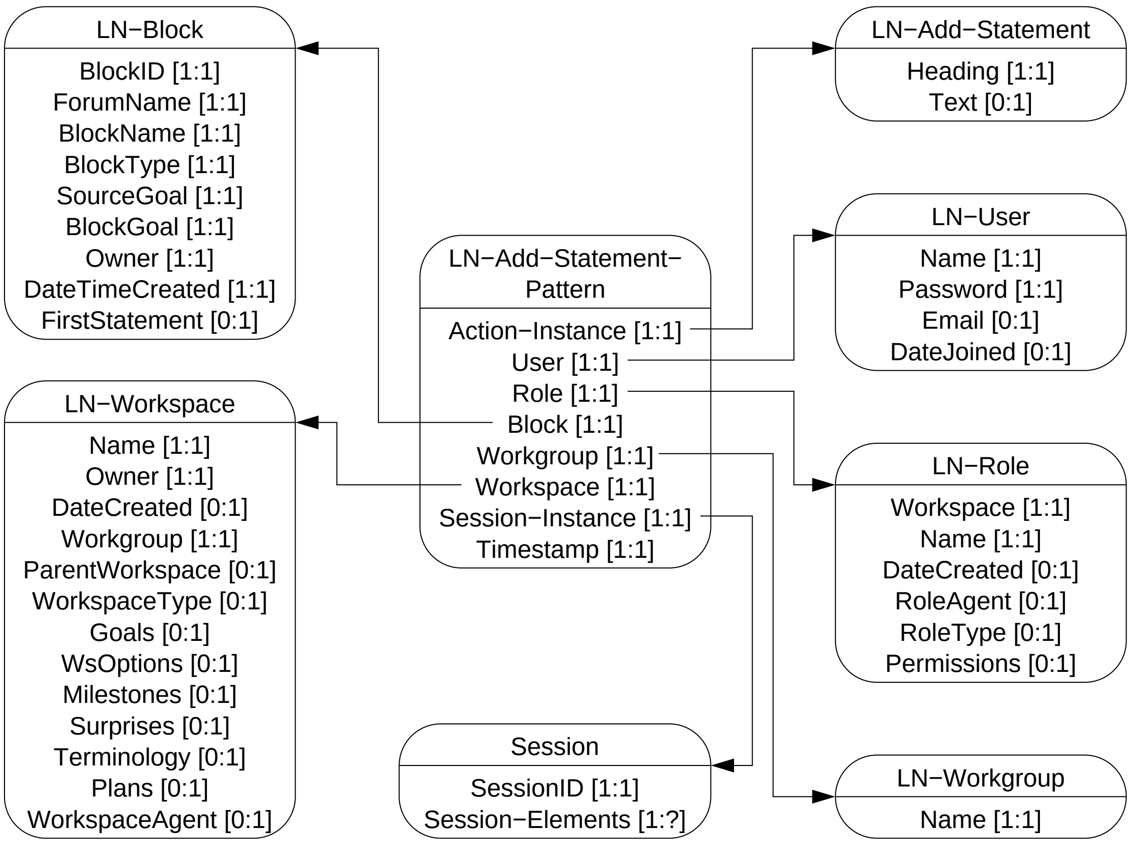


Figure 5.4: Action pattern class `LN-Add-Statement-Pattern` and related action and object classes

The above specifications of one object, one action, and one action pattern are illustrative of the specifications of all other of LIVENET's system-level concepts, which are specified in the same manner. Once all these concepts have been specified, modeling of the base level is complete.

5.1.3 Step 3: Modeling the User Level

Modeling of the user level proceeds in the same manner as modeling of the system level. The first step is the identification of objects, actions, and action patterns, followed by their specification. However, unlike on the system level, which is the base level, modeling of the user level may not have to model every concept on this level in detail. Indeed, some concepts on this level may be identical to those on the lower level. Others may resemble those on the system level, but may differ only in some details, and may thus need to be modified for this level. However, some concepts may be entirely new on this level and thus need to be fully specified.

5.1.3.1 Step 3.1: Identifying Objects

In LIVENET, user-level objects correspond in large part to those on the system level. However, the objects on this level are those as seen through the user view, i.e. as they are perceived by the user. For some object types, this view does not necessarily correspond to that on the system level. For instance, for implementation reasons the system level has three objects to capture different aspects of a discussion forum (objects Discussion, Forum, and Block). In the user view, however, no such separation exists, and a discussion forum is therefore a single object. Identifying the objects on this level thus consists of examining the collaboration system from the user's point of view and determining what objects are present in this view. These objects then need to be compared with those on the system level to determine if they are unchanged or modified.

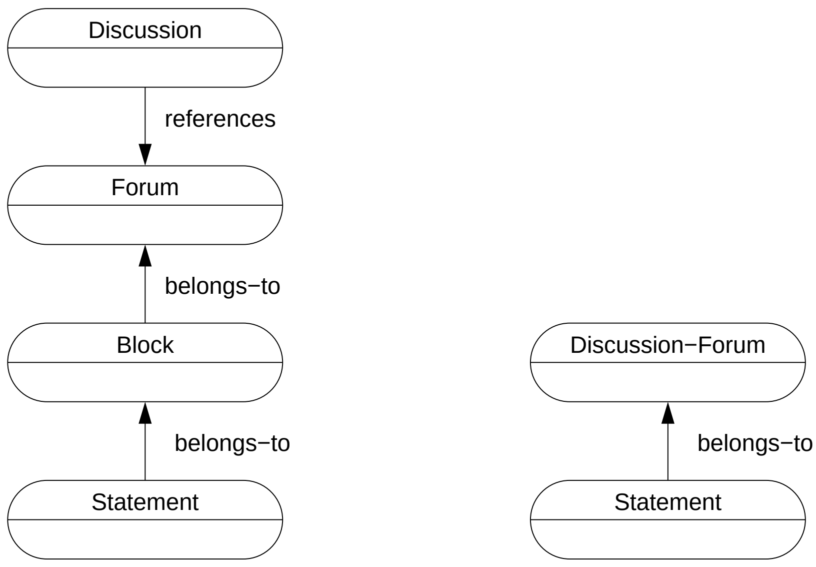
Identifying unchanged objects

The following ten system-level objects are unchanged on the user level:

- Action
- Background
- Document
- Message
- Message-Rule
- Message-Type
- Role
- User
- Workgroup
- Workspace

Identifying modified objects

As mentioned earlier, the objects related to discussions differ to some extent between the system and user levels. On the system level, discussions were represented as a discussion object, with an associated forum, which in turn has an associated block, which in turn contains a number of discussion statements. This arrangement is shown in Figure 5.5 (a).



(a) System-level discussion (b) User-level discussion

Figure 5.5: Different views of discussions on system and user levels

On the user level, i.e. as perceived through the user view, discussions are much simpler: to the user there are simply discussion forums consisting of discussion statements. This arrangement is represented in Figure 5.5 (b). Thus the objects Discussion, Forum, and Block are replaced with a single object Discussion-Forum. The object Statement, on the other hand, remains, but it now contains a reference to the Discussion-Forum which it belongs to, whereas on the system level it referred to the Block it belonged to.

Thus the modified concept on this level is the object Statement. Although the meaning of this concept remains unchanged, its details differ slightly from the level below.

Identifying new objects

As the preceding discussion mentioned, the objects Discussion, Forum, and Block are replaced by the object Discussion-Forum. This therefore constitutes the only new concept on this level:

Discussion-Forum:	a forum for the posting of discussion statements

In summary, of the 18 objects represented in the system-level part of the ontology, ten remain unchanged on the user level, and one is slightly changed. Seven objects, most of which are “internal” to the operation of LIVENET, and which thus do not appear in the user view of the system, are omitted from the user level; these are: Block, Discussion, Forum, Notification-Rule, Object, Participant, and User-Group. On the other hand, one new object, Discussion-Forum, has been introduced.

5.1.3.2 Step 3.2: Identifying Actions

User-level actions are primarily identified through exploration of the user interface: an interface function that can be performed by the user constitutes a user-level action. Thus by exhaustively exploring every function in the user interface, the set of actions can be determined.

As with objects, the user level's actions too differ from the system level. However, in the case of actions the difference is greater, as the commands which form the set of actions on the system level are all "internal", i.e. they are not exposed to the user view. While there is some correspondence between certain user-level and system-level actions, most user-level actions correspond not just to one, but to a sequence of several system-level actions. Moreover, the number of action attributes which are associated with a user-level action usually differs from the corresponding one on the system level, as some of these are too detailed to be of interest at this level, and are thus omitted. As a result, the set of user-level actions is for the most part disjoint from the corresponding set of system-level actions.

All in all, LIVENET has following 73 user-level actions:

Action Name	**Description**
Add-Discussion-Forum-Notification	Add a notification email address to a discussion forum
Add-Background	Add a background to a workspace
Add-Document	Add a document to a workspace
Add-User-To-Workgroup	Add a new user to a workgroup
Add-User-To-Workgroup-And-Workspace	Add a new user to a workgroup and a workspace
Add-User-To-Workspace	Add a user to a workspace
Add-Workflow-Rule	Create a workflow rule
Assign-Action	Assign an action to a user
Assign-Background	Assign a background to a user
Assign-Discussion-Forum	Assign a discussion forum to a user
Assign-Document	Assign a document to a user
Copy-Workspace	Copy a workspace

continued...

Action Name	**Description**
Create-Action	Create an action
Create-Discussion-Forum	Create a discussion forum
Create-Message-Type	Create a message type
Create-Role	Create a role
Create-Subworkspace	Create a workspace under the current one
Create-Workgroup	Create a workgroup
Deassign-Action	Deassign an action from a user
Deassign-Background	Deassign a background from a user
Deassign-Discussion-Forum	Deassign a discussion forum from a user
Deassign-Document	Deassign a document from a user
Delete-Action	Delete an action from a workspace
Delete-Background	Delete a background from a workspace
Delete-Current-Workspace	Delete the current workspace
Delete-Discussion-Forum	Delete a discussion forum from a workspace
Delete-Document	Delete a document from a workspace
Delete-Message	Delete a message from a workspace
Delete-Message-Rule	Delete a message rule from a workspace
Delete-Message-Type	Delete a message type from a workspace
Delete-Other-Workspace	Delete a workspace other than the current one
Delete-Role	Delete a role from a workspace
Delete-Workgroup	Delete a workgroup
Edit-Discussion-Statement	Modify a discussion statement
Edit-Role-Description	Edit the description of a role
Edit-Role-Permissions	Edit the permissions of a role
Edit-Workspace	Edit the properties of a workspace
Enter-User-Administration	Invoke the user administration function
Enter-Workspace	Enter a workspace
Invoke-Action	Invoke an action
List-Workflow-Rules	Show a list of workflow rules
Login	Login to the system
Logoff	Logoff from the system
Open-Background	Open a background
Open-Discussion-Forum	Open a discussion forum

continued...

Action Name	**Description**
Open-Discussion-Statement	Open a discussion statement
Open-Document	Open a document
Open-Message	Open a message
Open-Subworkspace	Open a workspace under the current one
Post-Discussion-Statement	Post a statement to a discussion forum
Refresh-Action-List	Redisplay the action list
Refresh-Background-List	Redisplay the background list
Refresh-Discussion-List	Redisplay the discussion statement list
Refresh-Document-List	Redisplay the document list
Refresh-Message-List	Redisplay the message list
Refresh-Participant-List	Redisplay the participant list
Refresh-Role-List	Redisplay the role list
Refresh-Subworkspace-List	Redisplay the subworkspace list
Refresh-Workflow-Rule-List	Redisplay the workflow rule list
Remove-Discussion-Forum-Notification	Remove a notification email address from a discussion forum
Remove-User-From-Workgroup	Remove a user from a workgroup
Remove-User-From-Workspace	Remove a user from a workspace
Replace-Background	Replace a background with a new one
Replace-Document	Replace a document with a new one
Send-Email	Send an email
Send-Message	Send a message
Switch-Workspace	Switch to another workspace
Update-User-Info	Update user information
Upload-Background	Upload a background
Upload-Document	Upload a document
View-Incoming-Messages	Display a list of received messages
View-Workflow-Rule-List	Display a list of workflow rules
View-Subworkspace-List	Display a list of subworkspaces

Identifying unchanged actions

The following four system-level actions are unchanged on the user level:

- Create-Workgroup
- Delete-Workgroup

- Logoff
- Send-Email

Identifying modified actions

The following two system-level actions are modified on the user level:

- Delete-Message
- Login

Although the semantics of both of these actions are the same as on the system level, they have slightly different or fewer action attributes on this level. For instance, while on the system level the Login action includes a user-id and password, on the user level the password is not included among its action attributes.

Identifying new actions

All remaining 67 actions identified above are new on this level.

Thus, in summary, of the 70 system-level actions four remain unchanged on this level, two exist in modified form on this level, and the remaining 64 actions do not exist on the user level. On the other hand, 67 user-level actions, based on combinations of system-level actions, are new on this level.

5.1.3.3 Step 3.3: Identifying Action Patterns

As on the system level, for each of the actions identified above, there is one corresponding action pattern, since each action again has structurally only one possible action context. The basic components of the action context are the same as before:

Subject: User, Role

Referent: depends on the action involved

Location: Workgroup, Workspace, Session

Time: timestamp of the action

Once the precise action context is identified for each of the 73 actions on the user level, the 73 user-level action patterns too can be identified.

Identifying unchanged action patterns

Corresponding to the actions identified above, the following four system-level action patterns are unchanged on the user level:

- Create-Workgroup-Pattern
- Delete-Workgroup-Pattern
- Logoff-Pattern
- Send-Email-Pattern

Identifying modified action patterns

The following two system-level actions are modified on the user level:

- Delete-Message-Pattern
- Login-Pattern

Identifying new action patterns

Action patterns for all remaining 67 actions identified above are new on this level.

Thus the situation for action patterns is the same as for actions: four action patterns are unchanged, two are modified, and 67 are new.

5.1.3.4 Step 3.4: Specifying Concepts

Having identified object, action, and action pattern concepts, these can now be specified. This again resembles the way that concepts were specified on the system level. On the user level, classes are specified as subclasses of the common classes `User-Lvl-Object`, `User-Lvl-Action`, and `User-Lvl-Action-Pattern`. Below, this is illustrated for unchanged, modified, and new concepts.

Specifying unchanged concepts

Static and dynamic entities, i.e. objects, actions, and action patterns, which remain unchanged across levels do not need to be re-specified on the higher level. However, the original specification of the concept on the lower level created its class as a subclass of that level's class of objects (e.g. an object on the system level is defined as a subclass of

Sys-Lvl-Object). This marked the class as belonging to, and being available on that level.

Therefore, in order for the same concept to be available on the higher level, its class specification needs to be extended by adding the higher level's class of objects as a superclass (this is the case of *unmodified one-to-one correspondence* mentioned in Section 4.2.3.1, p. 106). The resulting multiple inheritance of the class marks it as belonging to, and being available on, both levels.

For instance, the system-level class LN-Workgroup remains unchanged on the user level. Its specification can now be extended to include User-Lvl-Object as its superclass, as shown below (where the is-a slot in the third line now lists both of its superclasses):

```
(defclass LN-Workgroup "A conceptual entity grouping together users
                        and workspaces."
   (is-a Sys-Lvl-Object User-Lvl-Object)
   (role concrete)
   (single-slot Name
      (type STRING)
      (cardinality 1 1)))
```

The remainder of the class, however, is unchanged compared with its system-level specification. The definition of other classes corresponding to unchanged user-level objects are extended in the same manner, and the same applies to unchanged user-level actions and action patterns.

Specifying modified concepts

Modified concepts do not need to be specified anew in their entirety, as usually only some detail of the existing definition has changed. Therefore, specification of these concepts can be based on the existing definition and only needs to identify and modify those details that require change.

This raises the question how the modified class should be created. It is obvious that the original class cannot be changed to express the modified concept, as this would affect both levels and thus a modified class specification which would be valid on the higher level would be invalid on the lower level. Therefore the modified concept has to be specified as a new class. It would be easiest to create the new higher-level class as a subclass of the existing lower-level class, only overriding those slots which differ, but this too is problematic: doing so would imply that instances of the higher-level class are some kind of specialization of the lower-level class, and that some instances of the latter are also instances of the former. But this is not the case; instead, every instance of the higher-level class *corresponds* (but is not *equal*) to an instance of the lower-level class. Thus the specification of a modified concept has to be made as a new class, necessarily also with a different name than before, and directly sub-classing its own level's common base class.

An example of a modified concept is Statement which has, as mentioned earlier, the same meaning on the system and user levels, however differs slightly in detail between these two levels. Thus, while on the system level the class `LN-Statement`, which implements this concept, has a slot `Block` to indicate the Block and thus the Forum and Discussion it belongs to, on the user level these three concepts do not exist, as they have been replaced by the new concept Discussion-Forum. Instead, a different slot, `Disc-Forum`, is needed to identify the Discussion-Forum which the Statement belongs to. Copying the existing specification of `LN-Statement` (given above in Section 5.1.2.4, p. 150) as a base, the corresponding slot can be changed to produce the new class specification below:

```
(defclass LN-Discussion-Statement "A statement posted in a discussion
                                    forum."
   (is-a User-Lvl-Object)
   (role concrete)
   (single-slot Disc-Forum
      (type INSTANCE)
      (allowed-classes LN-Discussion-Forum)
      (cardinality 1 1))
   (single-slot StatementNo
      (type INTEGER)
      (cardinality 1 1))
   (single-slot Type
      (type STRING)
      (cardinality 0 1))
   (single-slot Originator
      (type INSTANCE)
      (allowed-classes LN-User)
      (cardinality 1 1))
   (single-slot ParentStmt
      (type INSTANCE)
      (allowed-classes LN-Discussion-Statement)
      (cardinality 0 1))
   (single-slot Heading
      (type STRING)
      (cardinality 1 1))
   (single-slot Text
      (type STRING)
      (cardinality 0 1))
   (single-slot ArtefactLink
      (type STRING)
      (cardinality 0 1))
   (single-slot DateTimeSent
      (type INTEGER)
      (cardinality 1 1)))
```

Note that the definition of the slot `ParentStmt` has also been modified as a result of the re-specification of this class for the user level, as this particular slot contains a self-reference, i.e. it now needs to refer to an instance of `LN-Discussion-Statement`.

Modified user-level actions and action patterns are specified in a similar manner. For instance, it was already mentioned that the user-level action Login differs slightly from the corresponding system-level action by omitting the password action attribute. Thus the specification of the user-level concept for this action is obtained by re-specifying the class for this level, in the same manner as for the specification of object `LN-Discussion-Statement` above.

Specifying new concepts

The specification of new concepts on higher levels in general parallels that of the specification of new concepts on the base level: the required classes have to be specified in full, from scratch. However, in some cases new concepts are related to existing lower-level concepts, and in these cases some parts of those existing definitions can flow into the new specifications.

For instance, as identified earlier, the user-level concept `LN-Discussion-Forum` is new on this level and thus needs to be specified in full. However, this concept is related to the concepts `LN-Discussion` and `LN-Forum`, and in fact some of the slots of those two classes are also needed in the new class `LN-Discussion-Forum`. Thus the class `LN-Discussion-Forum` can reuse the specification of those slots, as shown below:

```
(defclass LN-Discussion-Forum "A forum for the posting of discussion
                              statements."
   (is-a User-Lvl-Object)
   (role concrete)
   (single-slot Name
      (type STRING)
      (cardinality 1 1))
   (single-slot Creator
      (type INSTANCE)
      (allowed-classes LN-User)
      (cardinality 1 1))
   (single-slot DateCreated
      (type INTEGER)
      (cardinality 0 1)))
```

Here, the specification of the slot `Name` is taken from class `LN-Discussion`, while specifications of slots `Creator` and `DateCreated` are taken from class `LN-Forum`.

New user-level actions need to be specified in full too, while again building on some of the specification of system-level classes. An example of a class representing the user action Post-Discussion-Statement is shown below:

```
(defclass LN-Post-Discussion-Statement "An action that posts a
                             statement to a discussion forum."
   (is-a User-Lvl-Action)
   (role concrete)
   (single-slot Heading
      (type STRING)
      (cardinality 1 1))
   (single-slot Text
      (type STRING)
      (cardinality 0 1)))
```

With the inclusion of action context, as in the case of the system level, the corresponding action pattern can be specified as follows:

```
(defclass LN-Post-Discussion-Statement-Pattern "An action pattern
                   that posts a statement to a discussion forum."
   (is-a User-Lvl-Action-Pattern)
   (role concrete)
   (single-slot Action-Instance
      (type INSTANCE)
      (allowed-classes LN-Post-Discussion-Statement)
      (cardinality 1 1))
   (single-slot User
      (type INSTANCE)
      (allowed-classes LN-User)
      (cardinality 1 1))
   (single-slot Role
      (type INSTANCE)
      (allowed-classes LN-Role)
      (cardinality 1 1))
   (single-slot Discussion-Forum
      (type INSTANCE)
      (allowed-classes LN-Discussion-Forum)
      (cardinality 1 1))
   (single-slot Workgroup
      (type INSTANCE)
      (allowed-classes LN-Workgroup)
      (cardinality 1 1))
   (single-slot Workspace
      (type INSTANCE)
      (allowed-classes LN-Workspace)
      (cardinality 1 1))
   (single-slot Session-Instance
      (type INSTANCE)
      (allowed-classes Session)
      (cardinality 1 1))
   (single-slot Timestamp
      (type INTEGER)
      (cardinality 1 1)))
```

This specification resembles that of the system-level class `LN-Add-Statement-Pattern`, with the exception of the slots `Action-Instance` and `Discussion-Forum` which now refer to classes that are new on the user level[2].

[2]However, as the discussion about the mapping of concepts across levels below will show, the concepts differ more than in only this respect.

Once all user-level objects, actions and action patterns have been specified in this manner, the specification of this level is complete, and the next step, mapping of concepts, follows.

5.1.4 Step 4: Defining Mappings Between System Level and User Level

The previous two steps have created specifications of concepts on the system level and user level, respectively. However, these concepts stand separate from each other. Thus it is, for instance, not possible to tell from the specification that the user-level object `LN-Discussion-Forum` is related to, and can be derived from, the system-level objects `LN-Discussion` and `LN-Forum`; or that the user-level action pattern `LN-Post-Discussion-Statement-Pattern` is related to, and can be derived from, the system-level action pattern `LN-Add-Statement-Pattern`. These relationships and mappings across the specified classes need to be explicitly defined in the current step, in order to enable the transformation of information from one level to the next higher level.

Section 4.2.3 (p. 106) discussed three different types of relationships between concepts on adjacent levels, and how to map between them. The simplest type, *unmodified one-to-one correspondence,* was already encountered above in the specification of system-level concepts that are unchanged on the user level. Since the other two types of correspondences are similar (they both require the identification and specification of mappings of slots across classes), a single case is presented below which illustrates the mapping in detail for the action pattern class `LN-Post-Discussion-Statement-Pattern`.

5.1.4.1 Step 4.1: Identifying Source and Target Concepts and Attributes

The previous step identified and specified concepts on the user level. One of these was the action Post-Discussion-Statement. This is a new concept on the user level, which does not exist in that form on the system level. However, this action corresponds to the sequence of three system-level actions, namely Get-Block-Tree, Add-Statement, followed again by Get-Block-Tree. This is the example that was shown in Figure 3.8 on page 80. Now the mapping of these action concepts is shown.

To map from the system-level classes to the user-level class, first it is necessary to identify the source of each of the slots of the target class `LN-Post-Discussion-Statement`. There are two slots, and consequently two slot mappings:

No.	Source class	Source slot	Target slot
1	`LN-Add-Statement`	`Heading`	`Heading`
2	`LN-Add-Statement`	`Text`	`Text`

In this case, both slots of the target class are mapped from the same source class.

5.1.4.2 Step 4.2: Identifying Mapping Constraints

For the mapping of actions, the sequence of source actions being mapped to the target action constitutes a constraint. In this case, a sequence of three source actions needs to be mapped to a single target action:

Seq.No.	Source action	Target action
1	`LN-Get-Block-Tree`	
2	`LN-Add-Statement`	`LN-Post-Discussion-Statement`
3	`LN-Get-Block-Tree`	

5.1.4.3 Step 4.3: Specifying Mappings

Specifying slot mappings

Source classes need to be mapped to target classes according to the identified slot mappings. This means creating instances of the `Class-Mapping` and `Simple-Slot-Mapping` classes. The following are instances representing the mapping identified above:

```
([Class-Mapping_02] of Class-Mapping
   (Target-Class LN-Post-Discussion-Statement)
   (Slot-Map
      [Simple-Slot-Mapping_04]
      [Simple-Slot-Mapping_05]))

([Simple-Slot-Mapping_04] of Simple-Slot-Mapping
   (Target-Class LN-Post-Discussion-Statement)
   (Target-Slot [Heading])
   (Source-Class LN-Add-Statement)
   (Source-Slot [Heading]))

([Simple-Slot-Mapping_05] of Simple-Slot-Mapping
   (Target-Class LN-Post-Discussion-Statement)
   (Target-Slot [Text])
   (Source-Class LN-Add-Statement)
   (Source-Slot [Text]))
```

The first instance is of class `Class-Mapping`, identifying the target class, `LN-Post-Discussion-Statement`. It contains references to the two instances of class `Simple-Slot-Mapping` that follow. These in turn specify the slot mappings identified in Step 4.1 above.

Specifying action sequence mappings

For the mapping of actions, a sequence of actions on one level needs to be mapped to a single action on the next higher level. For this purpose, the class `Sequence-Mapping` was defined in Chapter 4. This class now needs to be instantiated in order to represent the mapping of the sequence identified in Step 4.2 above. The corresponding instance is shown below:

```
([Sequence-Mapping_01] of Sequence-Mapping
   (Mapping-Target LN-Post-Discussion-Statement)
   (Sequence-Elements
      LN-Get-Block-Tree
      LN-Add-Statement
      LN-Get-Block-Tree))
```

5.1.4.4 Step 4.4: Defining Mapping Functions

Having specified the mapping of slots, and the mapping of action sequences for the given target class, it is possible to determine how to construct instances of the target class when the action sequence is encountered in a session. As discussed in Section 4.2.4 (p. 118), this involves defining a *mapping function*.

For the given case of mapping to the user-level action `LN-Post-Discussion-Statement`, the mapping function `create-ln-post-discussion-statement` can be defined, as follows:

```
(deffunction create-ln-post-discussion-statement
  (?session ?pos)
  (if (and (instancep ?session)
           (eq (class ?session) Session)
           (integerp ?pos)
           (eq (class (nth$ ?pos
                        (slot-get ?session Session-Actions)))
                   LN-Get-Block-Tree)
               (eq (class (nth$ (+ ?pos 1)
                        (slot-get ?session Session-Actions)))
                   LN-Add-Statement)
               (eq (class (nth$ (+ ?pos 2)
                        (slot-get ?session Session-Actions)))
                   LN-Get-Block-Tree))
    then
      (bind ?addstmt (nth$ (+ ?pos 1)
                        (slot-get ?session Session-Actions)))
      (make-instance of LN-Post-Discussion-Statement
         (Heading (slot-get ?addstmt Heading))
         (Text (slot-get ?addstmt Text)))
      (return (+ ?pos 3))
    else
      (return ?pos)
  )
)
```

This function receives two input parameters, ?session which holds an instance of a session, and ?pos which identifies the first action, or starting position, in the session from where to start mapping actions. The function then performs some input validation and tests whether from the given starting position, a sequence of actions LN-Get-Block-Tree, LN-Add-Statement, and LN-Get-Block-Tree can be found. If all conditions are satisfied, an instance of LN-Post-Discussion-Statement is created with slot values taken from the corresponding slots of the instance of LN-Add-Statement occurring in the session. Finally, the function returns the position in the session after the matched action sequence: if the sequence was successfully matched, the new position is the old position plus three (for the three actions in the matched sequence), to give a new starting position for a subsequent matching of the remaining action sequences in the session; if the sequence was not matched, the old position is returned so that another function can be called to attempt to match it. In this way, repeatedly cycling through all matching functions for a session will eventually map all system-level action sequences to user-level actions.

Mapping functions for other action mappings are created in the same manner: slot mappings and action sequences for a given mapping of actions are identified and specified, then a mapping function is defined.

When all such functions have been specified, the step of mapping between system level and user level is complete.

5.1.5 Step 5: Modeling the Collaboration Level

On the collaboration level, larger units of activity involving two or more users, as well as involving other objects, are modeled. The main distinguishing feature of this level compared with the user level is that actions on this level are less detailed, and are for the most part summaries of those on the level below. For instance, the example of group discussion shown in Figure 3.10 (p. 84) is a case in point: the collaboration-level action Group-Discussion is a summary of multiple instances of multiple kinds of user-level actions. These and other group actions are modeled on this level. As before, the modeling of this level is performed by modeling its objects, actions, and action patterns.

5.1.5.1 Step 5.1: Identifying Objects

Objects on the collaboration level are those as seen through the collaboration-level view. As mentioned before, this level differs from the one below in that it considers collaborative actions, i.e. actions involving multiple people. The objects appearing in the collaboration-level view are thus largely the same as the ones appearing in the user-level view. However, some user-level objects are too detailed for consideration at this level, and are thus not represented. For instance, while Discussion-Forum is an object which is involved in collaborative activity between two or more users, the discussion that takes place within it is considered only at an aggregated level, not at the level of individual instances of Statement. Thus, discussion statements do not appear in the collaboration-level view. Similarly, the objects Message, Message-Type and Message-Rule are too detailed constructs, and are only considered collectively in the form of a Message-Channel object.

Identifying unchanged objects

The following eight user-level objects remain unchanged on this level:

- Action
- Background
- Discussion-Forum

- Document
- Role
- User
- Workgroup
- Workspace

Identifying modified objects

No objects from the user level are modified on this level.

Identifying new objects

The object Message-Channel is new on this level, encapsulating the three separate user-level objects Message, Message-Rule and Message-Type:

Message-Channel: a communication channel from a user to a role

5.1.5.2 Step 5.2: Identifying Actions

Actions on the collaboration level involve multiple people. In the case of LIVENET, where users always perform actions in a certain *role*, collaborative actions may thus involve: (1) multiple users occupying the same role; or (2) multiple users each occupying a different role; or (3) a combination of the two.

Identifying the set of collaborative actions on this level is performed by examining which user-level action concepts *connect* two or more users. Users can be connected in two different ways: (1) *direct* connection, where a user-level action performed by one user is directed at another user; or (2) *indirect* connection, where a user-level action performed by one user affects an object which is subsequently accessed by another user, thus *mediating* the effect of one user's actions on the other user. Collections of such user-level actions constitute collaboration-level actions. Therefore, there is no one-to-one correspondence between user-level actions and collaboration-level actions. For this reason there are no unchanged or modified actions on this level, instead all actions on this level are new.

Actions directly connecting users

For an action that directly connects users, there has to exist at least one user-level action that is explicitly directed at another user (or role). In the case of LIVENET, there are two such actions, both related to communication: Send-Email and Send-Message. Both of these are initiated by one user and send a message to another user, either by external email, or internally according to a workflow rule. Upon receipt of an internal message, the message is read by the receiver through the user-level action Open-Message[3]. The corresponding collaboration-level action is Message-Exchange: for each pair of users, any user-level actions of type Send-Email, Send-Message, or Open-Message involving those two users as either sender or recipient, taken collectively constitute an instance of Message-Exchange.

Actions indirectly connecting users

Actions indirectly connecting users involve objects accessed by other users. Identifying these actions consists of identifying user-level actions which operate on objects that are subsequently accessed by other users using the same or related user-level actions.

An example of this was mentioned above: group discussion. A user performs the Post-Discussion-Statement action, thereby posting a statement to a discussion forum. Subsequently, another user performs the Open-Discussion-Statement action, reading the statement that was posted earlier. This user can then in turn post statements which are read by other users, etc. In this case, the related user-level actions are Post-Discussion-Statement and Open-Discussion-Statement. The connecting objects are the discussion forum and the statements posted in it. The collaboration-level action Group-Discussion thus corresponds to the aggregation of all the related user-level actions for posting and opening discussion statements in a given discussion forum.

Another example is related to the exchange of documents. Documents are frequently used in collaborative activity to share information among a group of users. The related user-level actions involved are Add-Background, Add-Document, Open-Background, Open-Document, Replace-Background, Replace-Document, Upload-Background, and Upload-Document. The connecting objects are the documents and backgrounds being added, opened, replaced, or uploaded in a workspace. The collaboration-level action Artefact-Exchange corresponds to the aggregation of these user-level actions within a given workspace.

Two other collaboration-level actions exist: Workspace-Setup, which refers to the

[3]External email messages are opened with an external email application, of which no record exists in the LIVENET log, thus they cannot be considered within the Information Pyramid.

collaborative configuration and setup of a workspace, aggregating such user-level actions as Create-Role, Add-User-To-Workspace, Create-Discussion, etc. The other collaboration-level action is Content-Management, which refers to the management of content such as documents and backgrounds in a workspace, aggregating such user-level actions as Assign-Document, Deassign-Document, Delete-Document, etc.

In total there are thus following five basic collaborative actions in LIVENET:

Action Name	**Description**
Artefact-Exchange	Exchange of documents and/or backgrounds among a group of users
Content-Management	Rights management and removal of documents and/or backgrounds, performed collaboratively by two or more users
Group-Discussion	Exchange of discussion statements among a group of users
Message-Exchange	Asynchronous direct communication between a pair of users through messages
Workspace-Setup	Creation, setup and configuration of a workspace, performed collaboratively by two or more users

This set of actions is small, much smaller than the corresponding sets on the lower levels. The reason for this is that the sets of actions on the levels below are *complete* in that they capture all possible actions on those levels, whereas on this level the set of actions is not complete. The issue of completeness of specification was first addressed in Section 4.4 (p. 125) in the context of visualization. When considering the system-level set of actions, for instance, it corresponds to the set of LIVENET commands, with one action for each command. For any given version of the system, this set of commands is finite and fixed, and thus the set of corresponding system-level actions is complete. Similarly for the user level: the set of actions here consists of all the functions provided to the user through the LIVENET user interface. Again, for any given version of the system, this set of functions is finite and fixed, and thus the set of corresponding user-level actions is complete. On the collaboration level, however, the set of actions consists of certain aggregations of certain subsets of user-level actions. Given different subsets and different aggregations, different collaboration-level actions can be defined.

Thus the set of actions on this level is not complete, but is *open-ended* and *extensible*, by identifying different basic types of actions, or different sub-types of basic action types.

5.1.5.3 Step 5.3: Identifying Action Patterns

The five collaboration-level actions identified above can be regarded as constituting a set of *basic* actions for this level, while leaving open the possibility of defining different *subtypes* of these actions. For instance, the Group-Discussion action simply refers to the posting and reading of statements in a discussion forum among a group of users. It does not specify in what manner different users participate in this group discussion. However, it is possible to distinguish between different *styles* of the Group-Discussion action based on, for example, the differences in use between different users. For instance, one style of group discussion could be *question and answer*, where one user or role posts statements that initiate new discussion threads (to ask questions), while another user or role posts replies to those statements (answering the questions). This action still matches the same basic characteristics of the Group-Discussion action (a group of users posting and reading discussion statements in a given discussion forum), but this style of discussion constitutes a *specialization*, or subtype of the basic group discussion action. Another specialization of the Group-Discussion action is one where the discussion forum is used by one role to post statements that are notices or announcements, while another role simply reads these statements without posting any statements of their own. Figure 5.6 shows this in the form of both an EMOO diagram and the structure of the concept represented: part (a) of the figure shows the basic action pattern Group-Discussion-Pattern, while part (b) shows the action pattern Notice-Board-Pattern.

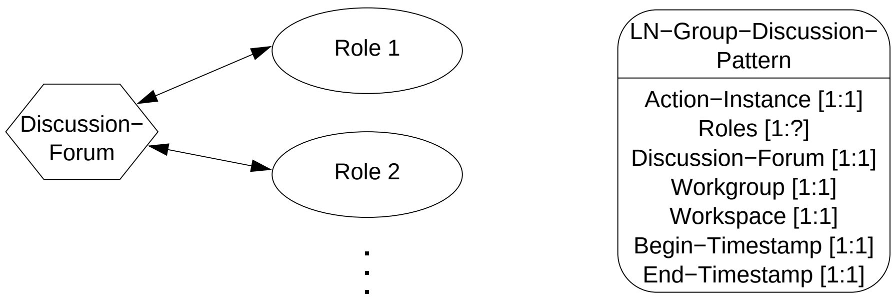
These two action patterns differ in that the first one, Group-Discussion-Pattern, provides for an arbitrary number of roles, while the second one, Notice-Board-Pattern, has exactly two roles: Poster and Reader, where the first one is the role that posts statements, and the second one is the role that reads statements. Furthermore, the Poster role performs both read and post actions (i.e. instances of the Open-Discussion-Statement and Post-Discussion-Statement actions), while the Reader role performs only read actions.

Other specializations of this and other action patterns may similarly exist, and can be identified by exploring the data collected from the collaboration system.

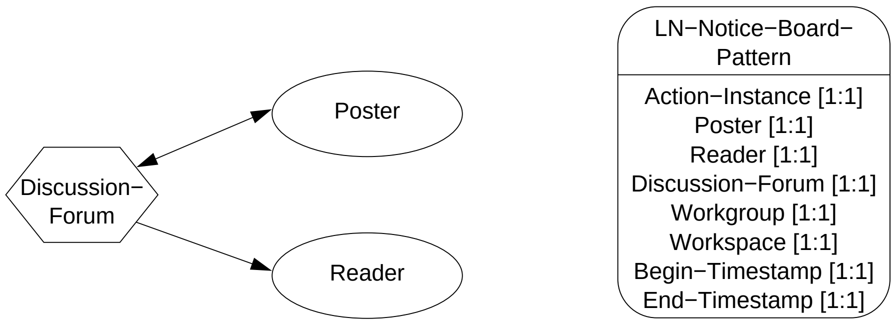
5.1.5.4 Step 5.4: Specifying Concepts

Having identified concepts, these can now be specified. As this parallels the specification of concepts on lower levels, it is omitted here.

Once again, when the specification of objects, actions, and action patterns is complete, the next step, mapping of concepts, can follow.



(a) Group-Discussion-Pattern



(b) Notice-Board-Pattern

Figure 5.6: Different types of action patterns based on action Group-Discussion

5.1.6 Step 6: Defining Mappings Between User Level and Collaboration Level

As with the previous mapping, it is again necessary to define how concepts on the two levels just modeled are related.

On the collaboration level, most objects were found to be unchanged from the user level, so that no mappings need to be defined for these. Only one new object was introduced (Message-Channel) which needs to be mapped, in the same manner as other objects on lower levels. The main focus of mapping on this level, however, is the mapping of actions. The mapping of the five basic collaboration-level actions is quite similar, and is illustrated below for the Group-Discussion-Pattern action concept.

5.1.6.1 Step 6.1: Identifying Source and Target Concepts and Attributes

Similar to the mapping on the lower levels, again the sources of each target slot in each class to be mapped need to be identified. On the collaboration level, however, the type of correspondence to the lower level is different than on the level below: here it is an *aggregation* of multiple instances of lower-level classes, as opposed to a direct correspondence of slot values across levels. Therefore not all target slots can be mapped from some given source slots, and some may instead need to be *computed*. For the class `LN-Group-Discussion-Pattern`, the mapping of slots is identified below[4]:

No.	**Source class**	**Source slot**	**Target slot**
1	`LN-Post-Disc-Stmt-Pat.` `LN-Open-Disc-Stmt-Pat.`	*all*(`Role`)	`Roles`
2	`LN-Post-Disc-Stmt-Pat.` `LN-Open-Disc-Stmt-Pat.`	`Discussion-Forum`	`Discussion-Forum`
3	`LN-Post-Disc-Stmt-Pat.` `LN-Open-Disc-Stmt-Pat.`	`Workgroup`	`Workgroup`
4	`LN-Post-Disc-Stmt-Pat.` `LN-Open-Disc-Stmt-Pat.`	`Workspace`	`Workspace`
5	`LN-Post-Disc-Stmt-Pat.` `LN-Open-Disc-Stmt-Pat.`	*min*(`Timestamp`)	`BeginTimestamp`
6	`LN-Post-Disc-Stmt-Pat.` `LN-Open-Disc-Stmt-Pat.`	*max*(`Timestamp`)	`EndTimestamp`

Some notes on a few special mappings: the first slot, `Roles`, a multislot, is mapped by aggregating all of the values (indicated by the aggregation function *all*) of the single-slot `Role` of the instances of the given source classes. The last two slots, `BeginTimestamp` and `EndTimestamp`, are also mapped through an aggregation by obtaining, respectively, the minimum and maximum values of the `Timestamp` slot from all instances of the two indicated source classes.

5.1.6.2 Step 6.2: Identifying Mapping Constraints

For the mapping to the `LN-Group-Discussion` action, the only constraint is that the instances of the source classes need to share the same values for the context attributes `Workgroup`, `Workspace` and `Discussion-Forum`. This can be expressed as follows:

[4]Note that in the table the names of source classes `LN-Post-Discussion-Statement-Pattern` and `LN-Open-Discussion-Statement-Pattern` have been shortened to `LN-Post-Disc-Stmt-Pat.` and `LN-Open-Disc-Stmt-Pat.`, respectively.

Constraint

```
(and (eq (slot-get ?PostStmt Workgroup)
         (slot-get ?OpenStmt Workgroup)
         ?Workgroup)
     (eq (slot-get ?PostStmt Workspace)
         (slot-get ?OpenStmt Workspace)
         ?Workspace)
     (eq (slot-get ?PostStmt Discussion-Forum)
         (slot-get ?OpenStmt Discussion-Forum)
         ?Discussion-Forum))
```

Here, `?Workgroup`, `?Workspace`, and `?Discussion-Forum` are variables referring, respectively, to instances of the workgroup, workspace, and discussion forum that form the shared context, while `?PostStmt` and `?OpenStmt` are variables ranging over all instances of the `LN-Post-Discussion-Statement-Pattern` and `LN-Open-Discussion-Statement-Pattern` classes, respectively.

5.1.6.3 Step 6.3: Specifying Mappings

The previously identified source and target classes and slots need to specified in the form of a class mapping and multiple slot mappings. As on other levels, this again means creating instances of the `Class-Mapping` and `Simple-Slot-Mapping` classes. In addition, for a number of slots instances of `Aggregated-Slot-Mapping` need to be created. Details of this are omitted here for the sake of brevity.

5.1.6.4 Step 6.4: Defining Mapping Functions

Once again, having identified mappings of slots, as well as mapping constraints, it is possible to define a mapping function that can transform instances of the source classes to instances of the target class.

For the given case of mapping to the collaboration-level action pattern class `LN-Group-Discussion-Pattern`, the mapping function to be defined follows the same principles as the function `create-ln-post-discussion-statement` defined in step 4.4. Again, for the sake of brevity, details are omitted here.

When this and all other functions have been specified, the step of mapping between user level and collaboration level is complete.

5.1.7 Step 7: Modeling the Task Level

The task level contains information about larger units of activity, typically composed of a number of lower-level actions, configured in certain ways so as to support the performance of certain tasks. Once again, the modeling of this level considers objects, actions, and action patterns.

5.1.7.1 Step 7.1: Identifying Objects

As with the levels below, information at the task level is perceived through its own view, the task view. In this view, activity typically consists of not just one, but of several actions in combination. The objects involved in the activity remain the same as on the previous level, i.e.:

- Action
- Background
- Discussion-Forum
- Document
- Message-Channel
- Role
- User
- Workgroup
- Workspace

Thus, none of the objects on the task level are modified or new.

5.1.7.2 Step 7.2: Identifying Actions

Actions on the task level consist of combinations of actions from lower levels, for the most part from the collaboration level. These are combined in such a way as to facilitate the performance of certain tasks. Being combinations of lower-level actions, these actions are thus all new on this level.

Identifying the set of tasks on this level is performed by examining how collaboration-level actions are *connected* with each other. When the same objects, such as roles, documents, discussion forums, etc. exist in two or more collaboration-level actions, they

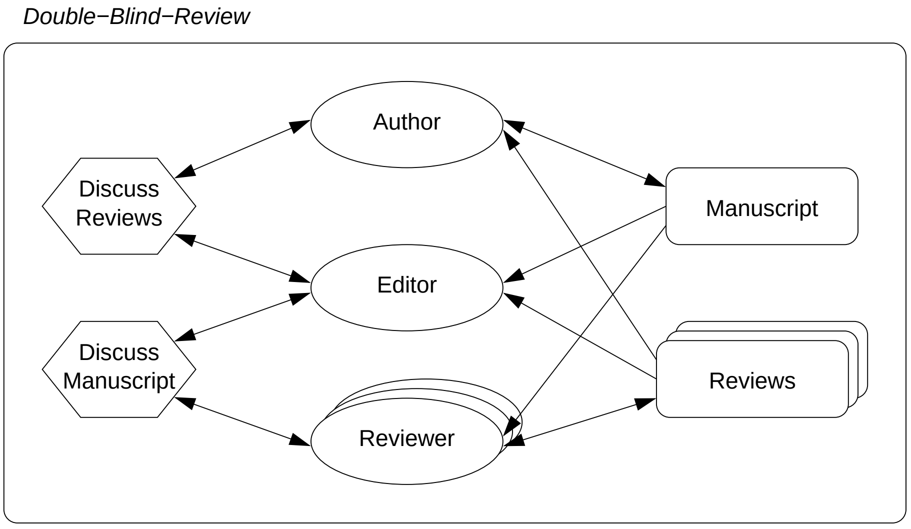


Figure 5.7: EMOO diagram of the Double-Blind-Review task-level action belonging to the publishing domain

are considered to be connected. An example of this was shown in Figure 3.11 (p. 86): collaboration-level actions Group-Discussion and Document-Sharing are connected, because the roles Coordinator and Writer exist in both. "Plugging together" such connected action patterns from the collaboration level produces the resulting task-level action pattern.

What was mentioned on the collaboration level about the open-endedness of the set of actions is even more so the case on the task level. Because any new combination of lower-level action patterns produces a new task-level action pattern, the number of such action patterns is virtually unlimited. However, unlike the level below which contained general-purpose action patterns such as for group discussion, document exchange, or peer-to-peer messaging, the action patterns on the task level usually are more or less *domain-specific*. The example of Figure 3.11, for instance, belongs to the domain of report writing, which is still a fairly generic domain. However, other task-level action patterns can belong to far more specialized domains. For instance, the domain of publishing involves tasks related to manuscript preparation. A task in this domain could be Manuscript-Layout or Chapter-Review. The configuration of these tasks is specific to the publishing domain: they involve domain-specific roles and artefacts, combined in a domain-specific way.

An example is the task of reviewing a manuscript in preparation for publication. A domain-specific form of review is the *double-blind review* where authors and reviewers are kept unaware of each other's identities. This is illustrated by the EMOO diagram shown in Figure 5.7. The task involves roles specific to the domain: Author, Editor, and

Reviewer; as well as artefacts: Manuscript and Reviews. The double-blind review is realized through two separate discussion forums: Discuss Manuscript for the editor and reviewer roles to discuss a manuscript submitted by an author; and Discuss Reviews for the editor and author roles to discuss any reviews submitted by the reviewers. Specific access configurations complete the setup of the action pattern: all roles have post/read access to the discussion forums, and read access to the three artefacts. Only the author role, however, has write access to the manuscript, while only the reviewer role has write access to the reviews. This particular combination of lower-level objects and actions thus constitutes a task-level action pattern specific to the domain of publishing. Other task-level action patterns from the same and other domains are made up in a similar fashion. The specific set of task-level action patterns for a given system and a given domain has to be gleaned from usage data collected from the system. The use of information visualization for the identification of such action patterns can greatly facilitate this process, as was discussed in Section 4.4.

5.1.7.3 Step 7.3: Identifying Action Patterns

Actions on the task level, being *combinations* of several lower-level actions rather than *aggregations* (as was the case on the lower levels), are not performed by a single user or at a single time. Instead they extend over a period of time and involve multiple users and/or roles. For this reason, the kind of information associated with the actions themselves, such as action attributes, is very limited, and most information belongs to the action context. Therefore, action patterns involving these actions contain most or all of the information associated with the tasks represented. Thus for the task represented in Figure 5.7, no action attributes can be identified, while all the relevant components, namely the collaboration-level action patterns involved (instances of Group-Discussion-Pattern and Artefact-Exchange-Pattern) as well as information about location and time are kept in a corresponding action pattern, Double-Blind-Review-Pattern. Figure 5.8 shows the structure of the class `LN-Double-Blind-Review-Pattern` representing this task-level action pattern.

Just as with this example, likewise for other actions their action patterns need to be identified so as to capture the relevant information.

5.1.7.4 Step 7.4: Specifying Concepts

Having identified concepts, these can now be specified. As this parallels the specification of concepts on lower levels, it is omitted here.

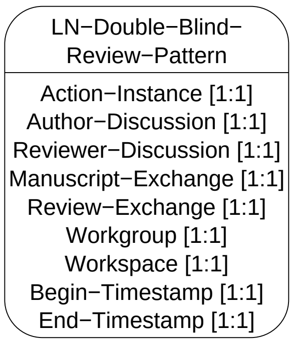


Figure 5.8: Task-level action pattern class `LN-Double-Blind-Review-Pattern`

Once again, when all objects, actions, and action patterns have been specified, the modeling of the task level is complete, and the mappings from collaboration to task level can be defined.

5.1.8 Step 8: Defining Mappings Between Collaboration Level and Task Level

As with the previous mapping, it is again necessary to define how concepts on the two levels just modeled are related.

In the case of the task level, no object concepts need to be mapped as all objects remain unchanged from the level below. Action and action pattern concepts, on the other hand, being all new on this level, require mapping.

5.1.8.1 Step 8.1: Identifying Source and Target Concepts and Attributes

Just as with previous mappings, again the sources of any slots in target classes need to be identified. To map to a task-level action pattern, its constituent collaboration-level action patterns need to be identified. Often, entire instances are mapped from the lower level to a slot on the higher level. This is illustrated below for the mapping to target class `LN-Double-Blind-Review-Pattern`[5]:

No.	Source class	Source slot	Target slot
1	`LN-Group-Disc-Pat.`	*instance*()	`AuthorDiscussion`

continued...

[5]Note that in the table the names of source classes `LN-Group-Discussion-Pattern` and `LN-Artefact-Exchange-Pattern` have been shortened to `LN-Group-Disc-Pat.` and `LN-Artef-Exch-Pat.`, respectively.

No.	Source class	Source slot	Target slot
2	`LN-Group-Disc-Pat.`	*instance*()	`ReviewerDiscussion`
3	`LN-Artef-Exch-Pat.`	*instance*()	`Manuscript-Exchange`
4	`LN-Artef-Exch-Pat.`	*instance*()	`Review-Exchange`
5	`LN-Group-Disc-Pat.` `LN-Artef-Exch-Pat.`	`Workgroup`	`Workgroup`
6	`LN-Group-Disc-Pat.` `LN-Artef-Exch-Pat.`	`Workspace`	`Workspace`
7	`LN-Group-Disc-Pat.` `LN-Artef-Exch-Pat.`	*min*(`BeginTimestamp`)	`BeginTimestamp`
8	`LN-Group-Disc-Pat.` `LN-Artef-Exch-Pat.`	*max*(`EndTimestamp`)	`EndTimestamp`

Here, the first four slots are mapped from *instances* of the source classes to the target slots. The next two slots are straightforward mappings of slot values. Finally, the last two slots are mapped through aggregation by obtaining, respectively, the minimum and maximum values of the indicated timestamp values in the source classes.

5.1.8.2 Step 8.2: Identifying Mapping Constraints

For the mapping to the action pattern class `LN-Double-Blind-Review-Pattern`, the constraints that need to be satisfied are that the instances of the source classes need to share the same values for the action context attributes `Workgroup` and `Workspace`, and that the individual collaboration-level action patterns are connected through the shared roles `Editor`, `Author`, and `Reviewer`. This can be expressed as follows:

No.	Constraint

1	`(and (eq (slot-get ?GroupDisc1 Workgroup)`
	`         (slot-get ?GroupDisc2 Workgroup)`
	`         (slot-get ?ArtefExch1 Workgroup)`
	`         (slot-get ?ArtefExch2 Workgroup))`
	`     (eq (slot-get ?GroupDisc1 Workspace)`
	`         (slot-get ?GroupDisc2 Workspace)`
	`         (slot-get ?ArtefExch1 Workspace)`
	`         (slot-get ?ArtefExch2 Workspace)))`

	continued...

No.	**Constraint**
2	`(exists ?Role`
	`   (and (eq (slot-get ?Role Name) Editor)`
	`        (member$ ?Role (slot-get ?GroupDisc1 Roles))`
	`        (member$ ?Role (slot-get ?GroupDisc2 Roles))`
	`        (member$ ?Role (slot-get ?ArtefExch1 Roles))`
	`        (member$ ?Role (slot-get ?ArtefExch2 Roles))))`
3	`(exists ?Role`
	`   (and (eq (slot-get ?Role Name) Author)`
	`        (member$ ?Role (slot-get ?GroupDisc1 Roles))`
	`        (not (member$ ?Role (slot-get ?GroupDisc2 Roles)))`
	`        (member$ ?Role (slot-get ?ArtefExch1 Roles))`
	`        (member$ ?Role (slot-get ?ArtefExch2 Roles))))`
4	`(exists ?Role`
	`   (and (eq (slot-get ?Role Name) Reviewer)`
	`        (member$ ?Role (slot-get ?GroupDisc2 Roles))`
	`        (not (member$ ?Role (slot-get ?GroupDisc1 Roles)))`
	`        (member$ ?Role (slot-get ?ArtefExch1 Roles))`
	`        (member$ ?Role (slot-get ?ArtefExch2 Roles))))`

Here, `?GroupDisc1` and `?GroupDisc2` are variables ranging over instances of the `LN-Group-Discussion-Pattern` class, one for each of the two group discussions involved in this task-level action pattern. Similarly, `?ArtefExch1` and `?ArtefExch2` are variables ranging over instances of the `LN-Artefact-Exchange-Pattern` class, for the two artefact exchanges involved in this task-level action pattern. Finally, `?Role` is a variable ranging over instances of the `LN-Role` class.

The first constraint states that the two instances of `LN-Group-Discussion-Pattern` and of `LN-Artefact-Exchange-Pattern` must reside in the same workgroup and workspace. The second constraint requires the `Editor` role to be present in both group discussions and both artefact exchanges. The third constraint requires the `Author` role to be present in the first group discussion, and not in the second one, and also to be present in both artefact exchanges. Lastly, the fourth constraint requires the `Reviewer` role to be present in the second group discussion, and not in the first one, and again in both artefact exchanges. In this way, the requirement of mutual anonymity between author and reviewers can be ensured.

5.1.8.3 Step 8.3: Specifying Mappings

The previously identified source and target classes and slots need to be specified in the form of a class mapping and multiple slot mappings. As on other levels, this again means creating instances of the `Class-Mapping` and `Simple-Slot-Mapping` classes. For some slots, instances of `Aggregated-Slot-Mapping` and `Instance-Slot-Mapping` need to be created. Details of this specification are omitted here.

5.1.8.4 Step 8.4: Defining Mapping Functions

Once again, having identified mappings of slots, as well as mapping constraints, it is possible to define a mapping function that can transform instances of the source classes to instances of the target class. In this case, the mapping function needs to take instances of some of the source classes and place references to these instances in the target class `LN-Double-Blind-Review-Pattern`. Apart from this difference, however, the mapping function required resembles the example shown earlier in Step 4.4, and is omitted here for brevity.

Mapping functions for other action mappings need to be created in the same manner: slot mappings and constraints need to be identified and specified, then a mapping function needs to be defined.

When all such functions have been specified, the mapping step for the task level is complete.

5.1.9 Step 9: Modeling the Process Level

The process level is the highest level in the Information Pyramid, containing information about processes, the largest units of activity. Processes are collections of tasks and inter-task relationships. As before, modeling of this level considers objects, actions and action patterns.

5.1.9.1 Step 9.1: Identifying Objects

Once again, on the process level information is perceived through its own view, the process view. This is a high-level view in which only tasks appear, and in which the objects involved in those tasks are not of concern. Thus no objects need to be modeled at this level.

5.1.9.2 Step 9.2: Identifying Actions

As just mentioned, the only information appearing at this level is related to the combination of collaborative tasks into processes.

Process-level action patterns can be identified by examining information flows between tasks. An information flow exists, for example, when one task creates a document and another task subsequently accesses that document. Information flows such as these always involve two tasks, one being an information producer, the other an information consumer. The example in Figure 2.6 (p. 29) showed several instances of such information flows between tasks (represented by an artefact between two tasks and an arrow pointing from a task to it and from it to another task). By looking for pairs of tasks that are linked through information flows, and for the tasks linked from those tasks, recursively, it is possible to identify a whole network of tasks that are thus related.

5.1.9.3 Step 9.3: Identifying Action Patterns

As with the task level, the set of action patterns on this level is both open-ended and domain-specific: any number of action patterns can be defined for any combination of tasks into processes that serve the collaborative work of specific domains. Actual processes can be obtained from the collaboration data, and information visualization can help identify process-level action patterns within it, as was discussed in Section 4.4. For instance, the task-level action pattern Double-Blind-Review-Pattern may be identified as belonging together with other task-level action patterns in a process of manuscript preparation.

5.1.9.4 Step 9.4: Specifying Concepts

Once information on this level has been identified, the corresponding concepts can be specified. On the process level, the only concepts to be specified are related to dynamic information, all of which are new on this level. Thus no unchanged or modified concepts need to be specified.

Process-level action patterns are the new concepts of this level. These action patterns are mainly represented by the process's tasks, together with a few attributes representing the process's action context. Thus specifications of process-level actions and action patterns need to create classes that contain references to the tasks belonging to the process.

An example was the manuscript preparation process-level action pattern mentioned above, which can be specified as the class `LN-Manuscript-Preparation-Pattern`, shown in Figure 5.9. The class consists of four slots for four task-level action patterns that

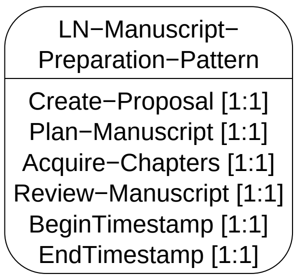


Figure 5.9: Process-level action pattern class `LN-Manuscript-Preparation-Pattern`

make up this process-level action pattern, and two action context attributes identifying this action pattern's time interval.

After all process-level actions and action patterns have been specified in this manner, the modeling of the process level is complete. Only the mapping from the task level to the process level remains in order to complete the Information Pyramid, which is discussed next.

5.1.10 Step 10: Defining Mappings Between Task Level and Process Level

As with the previous mapping, it is again necessary to define how concepts on the two levels just modeled are related.

In the case of the process level, there are no objects to be mapped as the process level view does not include objects. Process-level action patterns, however, being all new on this level, require mapping.

5.1.10.1 Step 10.1: Identifying Source and Target Concepts and Attributes

As before, target concepts need to be mapped from corresponding source concepts, by mapping slots across classes. In the case of process-level action patterns, most information consists of specifications of the collections of tasks involved, but usually a few slots of context information are also included which need to be mapped. An example of such a mapping is shown below, for class `LN-Manuscript-Preparation-Pattern`[6].

[6]Note that in the table the names of source classes have been shortened by abbreviating the `Pattern` suffix in the class name to `Pat`.

No.	Source class	Source slot	Target slot
1	`LN-Create-Proposal-Pat.`	*instance*()	`Create-Proposal`
2	`LN-Plan-Manuscript-Pat.`	*instance*()	`Plan-Manuscript`
3	`LN-Acquire-Chapters-Pat.`	*instance*()	`Acquire-Chapters`
4	`LN-Double-Blind-Review-Pat.`	*instance*()	`Review-Manuscript`
5	`LN-Create-Proposal-Pat.` `LN-Plan-Manuscript-Pat.` `LN-Acquire-Chapters-Pat.` `LN-Double-Blind-Review-Pat.`	*min*(`BeginTimestamp`)	`BeginTimestamp`
6	`LN-Create-Proposal-Pat.` `LN-Plan-Manuscript-Pat.` `LN-Acquire-Chapters-Pat.` `LN-Double-Blind-Review-Pat.`	*min*(`EndTimestamp`)	`EndTimestamp`

The `LN-Manuscript-Preparation` process-level action pattern involves four task-level action patterns: `LN-Create-Proposal-Pattern`, `LN-Plan-Manuscript-Pattern`, `LN-Acquire-Chapters-Pattern`, and `LN-Double-Blind-Review-Pattern`. The corresponding target slots are mapped from instances of the four task-level action patterns (indicated by the *instance*() keyword). The begin and end timestamp values of the process-level action pattern are mapped through aggregation, by obtaining the corresponding minimum and maximum values from these four task-level action patterns, respectively.

5.1.10.2 Step 10.2: Identifying Mapping Constraints

Given that the mapping to process-level action pattern `LN-Manuscript-Preparation-Pattern` is very simple, there are no mapping constraints.

5.1.10.3 Step 10.3: Specifying Mappings

The specification of mappings of process-level action patterns parallels that of the levels below; in the interest of brevity it is therefore not shown here.

5.1.10.4 Step 10.4: Defining Mapping Functions

Mapping functions need to receive a collection of task-level action patterns as their input and produce instances of process-level action patterns. This corresponds to the way that mapping functions on lower levels are implemented. For example, for the mapping to

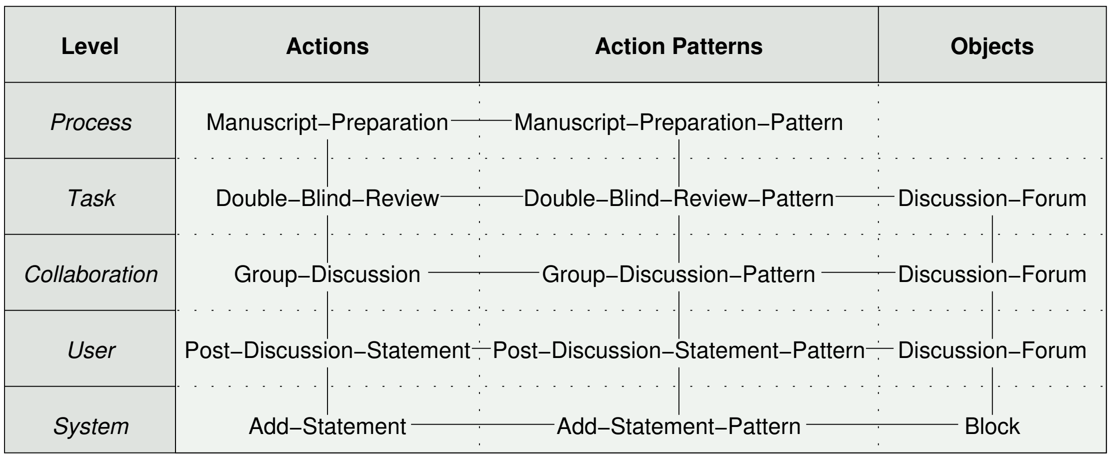


Figure 5.10: Horizontal and vertical links between some of the concepts in the LIVENET ontology on different levels of the Information Pyramid

process-level action pattern `LN-Manuscript-Preparation-Pattern`, a mapping function needs to receive four instances of task-level action patterns (one for each of the four classes involved). It then needs to produce an instance of the target class with references to these task-level action pattern instances, and needs to derive the overall begin and end timestamp values of the process-level action pattern from the instances of the task-level action patterns, as identified in Step 10.1 above.

Once all processes and their mappings have been specified, the Information Pyramid is complete, allowing the derivation of highest-level action patterns from lowest-level information.

5.1.11 Relationships Between Concepts in the Completed Ontology

Once the ontology of a particular collaboration system has been completely specified, such as the one for LIVENET that has been illustrated on the preceding pages, specifications of a large number of concepts exist. These concepts do not exist in isolation, but instead most of them are related to one another in different ways. For instance, some concepts may reference other concepts on the same level. Other concepts may be abstractions of one or more concepts on a lower level. Thus the completed ontology can be thought of as consisting of multiple levels (corresponding to the levels of the Information Pyramid), with *horizontal and vertical links* relating concepts within and across levels, respectively. This is illustrated in Figure 5.10.

The figure shows a few of the concepts that were discussed in the specification of the

LIVENET ontology on the preceding pages, with one example each of an action, an action pattern, and an object, on the five levels from system level up to the process level (recall that the system level was identified as the base level, which is why the infrastructure level is absent in the ontology; moreover, since on the process level there are no object concepts, the corresponding space in the figure is empty).

The horizontal lines linking concepts represent references from one to another concept. Thus, for example, the system-level concept Add-Statement-Pattern, an action pattern concept, is related to the action concept Add-Statement and the object concept Block on the same level, because the action pattern Add-Statement-Pattern involves the action Add-Statement, and an instance of Block is the referent of this action. Similar comments apply on the other four levels.

On the other hand, the vertical lines linking concepts represent relationships between sources and targets of mapped concepts, where the higher-level concept is an abstraction of the lower-level one. Thus, for example, the user-level action concept Post-Discussion-Statement is an abstraction of the system-level action concept Add-Statement, and is mapped from it. On the other hand, Post-Discussion-Statement is itself mapped to a higher-level concept, namely collaboration-level action concept Group-Discussion. The same applies to other concepts related through vertical links.

It can also be seen that some concepts appear on more than one level, such as the object concept Discussion-Forum which exists on the user, collaboration, and task levels. This is due to the fact that this concept appears unaltered in the respective views of those levels (i.e. the user, collaboration, and task views). It was seen that this concept is specified as belonging to all these levels by including the relevant levels' common classes as its superclasses.

Thus a completed ontology of a given collaboration system consists of a *matrix* of inter-related concepts that makes it possible to relate information obtained from collaboration data, and to derive progressively higher-level information from it.

5.2 Pattern Extraction from LIVENET Data

The preceding section has demonstrated the specification of an ontology of concepts related to the LIVENET collaboration system. The present section carries on from there to illustrate the use of this ontology in the extraction of patterns of virtual collaboration from data collected by the LIVENET collaboration system.

This process makes use of the LIVENET Workspace Visualizer, an information visualization tool developed for this purpose which assists in the identification of highest-

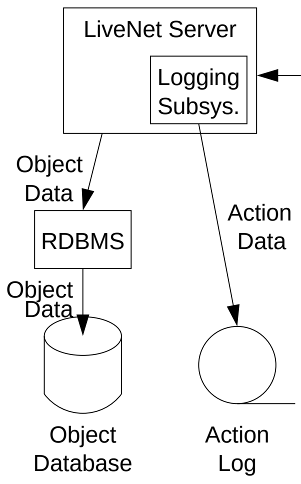


Figure 5.11: Data collection in LIVENET

level action patterns in the Information Pyramid. Details of this tool are given in Appendix A.

5.2.1 Data Collection

The extraction of patterns of virtual collaboration requires a body of data from which to extract the patterns. This in turn requires data collection facilities, and a source of data. Both of these are discussed below.

5.2.1.1 Data Collection Facilities

The extraction of patterns of virtual collaboration from a collaboration system requires that system to collect collaboration data. In the case of the LIVENET collaboration system, it was originally designed to only maintain a database of objects, but not to record any actions. In order to collect information on actions and thereby to make pattern extraction possible, a logging facility was added to LIVENET. This records details of all system-level actions taking place in the system.

Figure 5.11 shows the components involved in the data collection in LIVENET. LIVENET is a client-server system, with multiple clients connected to a single server. Clients send requests to the server, which in turn services these requests. The requests sent by LIVENET clients are actions to be performed by the server, usually (but not necessarily) involving LIVENET objects such as workspaces, documents, discussion forums, etc. When a LIVENET client (shown as the box on the top right) sends a request to perform an action to the LIVENET server (shown as the box on the top left), the server's logging subsystem records the action in an action log file. The LIVENET server then performs the action, which may result in the creation or modification of objects, which are stored in a database managed by a relational database management system with which the LIVENET server interacts. Thus information on both objects and actions can be obtained from LIVENET by accessing the object database and the action log.

5.2.1.2 Data Source

To extract patterns of virtual collaboration, data originating from actual users of the LIVENET system needed to be obtained. At the time of writing, most of the users of LIVENET are students and academics at the University of Technology, Sydney (UTS), as well as a few other universities. Students of the postgraduate-level course "Conducting Business Electronically" at UTS were selected as the source of data. These students were going to use the LIVENET system as part of their course, to design collaboration spaces and also to facilitate their own work of preparing group assignments, therefore they constituted a suitable user group for obtaining data. The students were asked for permission to use the data generated by their use of LIVENET. A total of 129 out of 232 students, about 56% of the class, consented to the use of their data. Data was collected for three months, from 10th August to 9th November 2000, which was the time period during which the students were using the LIVENET system. About half a million action records were collected in the action log, while the object database contained more than 600 workspaces created during the data collection period.

To illustrate the extraction of action patterns from this data, the data from one student group consisting of five members was chosen for pattern extraction. This group, designated as "Group 9", used LIVENET for facilitating the preparation of their group assignments. This work took place in one workspace, `cbe-group-09_Master`, referred to below as the group's "master workspace". The action log contained 2737 system-level actions performed by members of Group 9 in their master workspace during the data collection period.

The extraction and mapping of action patterns for Group 9 is shown in the remainder of this section; for details about the number of system-level actions per session, the number of sessions per day, and a count of different system-level actions performed, interested readers may refer to Appendix B .

5.2.2 Pattern Extraction

Once the data was collected, action patterns were extracted from it. This involved the use of both automated tools as well as manual extraction using database query facilities.

Moreover, the LIVENET Workspace Visualizer described in Appendix A was used in the identification of task-level action patterns. Identification and extraction of action patterns was guided by the previously specified ontology. The individual steps were:

1. *Import of action log to a database.* To facilitate the querying and manipulation of records in LIVENET's action log, the action log was imported into a relational database. A program specially written for this purpose created one row in a log database table for each log record contained in the action log file. The total number of such action log records for Group 9 was 2737. The program also performed some validation of the log data. These log records constitute the base-level (i.e. system-level) actions.

2. *Derivation of user-level actions.* Given the system-level actions in the log database table, user-level actions were derived according to the specifications of the LIVENET ontology, through simple matching of sequences of system-level actions. This was performed by another specially written program. A total of 909 user-level actions were derived from the 2737 system-level actions of Group 9.

3. *Derivation of action patterns at the collaboration level.* From the user-level actions, actions and action patterns at the collaboration level were derived. At this level, action patterns are more easily identifiable, and data volume is smaller, so this task was performed manually using SQL queries against the database of user-level action records. Once again, the higher-level actions and action patterns were derived according to the mappings specified in the LIVENET ontology.

4. *Identification of action patterns at the task level.* Given the collaboration-level action patterns, the LIVENET Workspace Visualizer was used to aid in the identification of task-level action patterns.

The records of actions mentioned above were complemented by records of objects contained in LIVENET's object database.

5.2.2.1 Mapping Action Patterns from System Level to User Level

The extraction and mapping of action patterns from the collected data is illustrated on one particular subset of data for one particular session involving one member of Group 9. The detailed action log records of this session (with session number 4228) are shown in the Appendix in Section B.2.4.

The session consists of a total of 38 system-level actions, as shown in the left half of the table below:

System-level actions		User-level actions	
1	`Getmyworkspaces`	1	`Login`
2	`Setworkspace`	2	`Enter-Workspace`
3	`Get-Led-Workgroups`		
4	`Get-Workspace-Tree`		
5	`Get-Role-Objects`		
6	`Get-Role-Objects`		
7	`Get-Role-Objects`		
8	`Get-Role-Objects`		
9	`Getroles`		
10	`Getparticipants`		
11	`Get-Role-Messages`		
12	`Get-Msg-Types`		
13	`Get-User-Email-Homepages`		
14	`Add-Object`	3	`Add-Document`
15	`Get-Role-Objects`		
16	`Add-Object`	4	`Add-Document`
17	`Get-Role-Objects`		
18	`Open-Object`	5	`Open-Document`
19	`Open-Object`	6	`Open-Document`
20	`Open-Object`	7	`Open-Document`
21	`Open-Object`	8	`Open-Document`
22	`Delete-Object`	9	`Delete-Document`
23	`Get-Role-Objects`		
24	`Add-Object`	10	`Add-Document`
25	`Get-Role-Objects`		
26	`Open-Object`	11	`Open-Document`
27	`Add-Object`	12	`Add-Document`
28	`Get-Role-Objects`		
29	`Add-Object`	13	`Add-Document`
30	`Get-Role-Objects`		
31	`Add-Object`	14	`Add-Document`
32	`Get-Role-Objects`		
33	`Get-Block-Tree`	15	`Open-Discussion-Forum`

continued...

System-level actions		**User-level actions**	
34	`Get-Block-Tree`	16	`Post-Discussion-Statement`
35	`Add-Statement`		
36	`Get-Block-Tree`		
37	`Normal-Close`	17	`Logoff`
38	`Logoff`		

For the given data of each of these actions, a corresponding system-level action pattern was extracted. For instance, from the sequence of system-level actions 34–36 (`Get-Block-Tree`, `Add-Statement`, `Get-Block-Tree`) the corresponding system-level action patterns `LN-Get-Block-Tree-Pattern`, `LN-Add-Statement-Pattern` and `LN-Get-Block-Tree-Pattern` were extracted, containing values for all slots of those action patterns. In the case of the `LN-Add-Statement-Pattern` action pattern, for example, several instances of objects are referenced, including one each of `LN-User`, `LN-Role`, `LN-Block`, `LN-Workgroup`, `LN-Workspace`, and `Session`, as well as the action `LN-Add-Statement` (this corresponds to the modeling of these concepts on the system level of the LIVENET ontology; refer to Figure 5.4 on page 155). Represented in the form of an EMOO diagram, the sequence of these three action patterns is depicted in Figure 5.12.

Having extracted system-level actions and action patterns from the source data, these were mapped to the user level according to the specified mappings in the LIVENET ontology. This mapping is shown in the right half of the table above. A sequence of one or more system-level actions were mapped to a single corresponding user-level action. The 38 system-level actions of session 4228 were thus mapped to 17 user-level actions. For example, the sequence of system-level actions 34–36 (`Get-Block-Tree`, `Add-Statement`, `Get-Block-Tree`) was mapped to user-level action 16 (`Post-Discussion-Statement`). An EMOO diagram showing an instance of the resulting user-level action pattern Post-Discussion-Statement-Pattern is shown in Figure 5.13.

5.2.2.2 Mapping Action Patterns from User Level to Collaboration Level

Once action patterns were mapped to the user level, the next step was to map user-level action patterns to the next-higher level, the collaboration level. Once again, this was performed based on the definitions of mappings contained in the LIVENET ontology.

To illustrate this for Group 9, the mapping of discussion-related action patterns shown above is continued here. As mentioned in Section 5.1.6, the user-level action patterns involved are `Post-Discussion-Statement-Pattern` and `Open-Discussion-Statement-Pattern`. Group 9 has performed a total of 118 such action patterns in 36 distinct

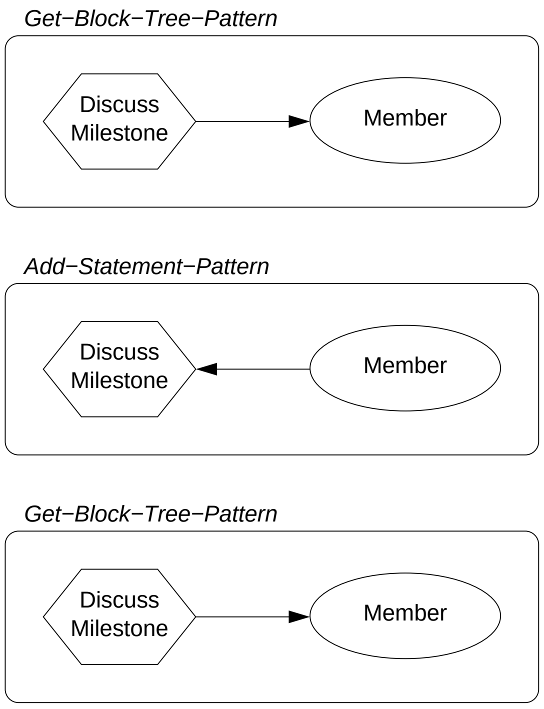


Figure 5.12: EMOO diagrams representing a sequence of instances of system-level action patterns Get-Block-Tree-Pattern, Add-Statement-Pattern, and Get-Block-Tree-Pattern, performed on discussion Discuss Milestone by a user of Group 9 occupying role Member

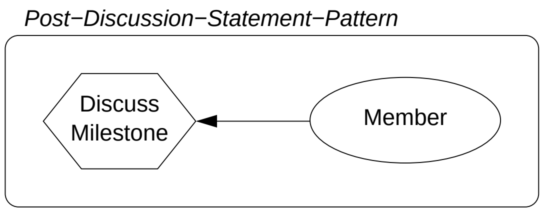


Figure 5.13: EMOO diagram representing an instance of user-level action pattern Post-Discussion-Statement-Pattern, performed on discussion Discuss Milestone by a user of Group 9 occupying role Member

sessions, namely 19 instances of the `Post-Discussion-Statement-Pattern` action pattern and 99 instances of the `Open-Discussion-Statement-Pattern` action pattern, as shown in the table below:

SessionId	**Posting actions**	**Opening actions**
527	1	1
673	2	5
981	0	2
1209	2	4
1316	0	4
1616	0	4
1627	0	3
1643	1	1
1701	1	9
1702	0	6
2039	0	1
2049	1	0
2362	0	1
2571	1	4
2732	0	4
2742	1	2
2862	2	2
3006	1	5
3190	0	5
3250	1	4
3960	0	4
4207	1	5
4228	1	0
4457	0	2
4478	0	1
4784	0	7
5232	0	2
5539	0	1
6527	0	2
7165	1	0
7919	0	1

continued...

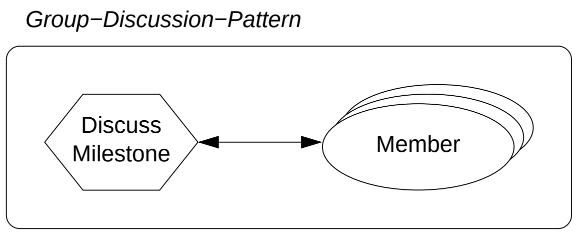


Figure 5.14: EMOO diagram representing an instance of collaboration-level action pattern Group-Discussion-Pattern, performed on discussion Discuss Milestone by the users of Group 9 occupying role Member

SessionId	**Posting actions**	**Opening actions**
8815	1	1
8870	0	2
8946	1	1
9218	0	1
10015	0	2
Total	19	99

These action patterns were all performed by the same role (Member) on the same discussion forum (Discuss Milestone) in the same workspace (the group's master workspace). Therefore, all of these action patterns were mapped to one and the same instance of collaboration-level action pattern `Group-Discussion-Pattern`. Represented in the form of an EMOO diagram, this action pattern is shown in Figure 5.14.

Besides these discussion-related action patterns, there were also document-related action patterns that took place in the master workspace of Group 9. The action patterns involved (and their count) over all sessions of Group 9 are `Open-Background-Pattern` (10), `Open-Document-Pattern` (179), `Upload-Background-Pattern` (1), and `Upload-Document-Pattern` (38). Because they all took place in the same workspace, and were performed by the same role (Member), they were all mapped to a single collaboration-level action pattern, namely `Artefact-Exchange-Pattern`. An EMOO diagram representing this action pattern is shown in Figure 5.15. The artefacts involved are shown as a multi-artefact, Report Components, made up of components of the various reports which the group was preparing.

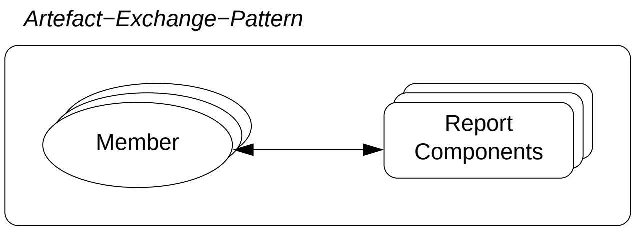


Figure 5.15: EMOO diagram representing an instance of collaboration-level action pattern Artefact-Exchange-Pattern, performed on multi-artefact Report Components by the users of Group 9 occupying role Member

5.2.2.3 Mapping Action Patterns from Collaboration Level to Task Level

The previous mapping step has established that there were primarily two collaboration-level action patterns that took place in the master workspace of Group 9: an instance of `Group-Discussion-Pattern` and an instance of `Artefact-Exchange-Pattern`. To assist in the mapping to the next-higher level, the task level, the LIVENET Workspace Visualizer was used.

A basic intra-workspace map of the master workspace of Group 9 is shown in Figure 5.16. It shows two roles: Member (at the top) and Owner (at the bottom). The Member role was occupied by six users: cbe-janr, cbe-shane, clehmann, ggold, imckean, and malibaba. The first five of these were the actual members of Group 9, while the sixth user was the course tutor assigned to this group. The Owner role was occupied by the user desnet-manager, which was the user id used by the course instructor.

The centre of the map is made up of 16 documents, all of which were accessible to both the Member and Owner roles. Moreover, one discussion forum, Discuss Milestones, was also accessible to both of these roles.

In the version of LIVENET from which the data was obtained, every newly created object is accessible to the role of the user who created it, as well as to the Owner role. This explains why all the documents and the discussion forum were accessible to both the Member and Owner roles. However, no action performed by the Owner role operating on documents or the discussion forum was recorded, and therefore this role can be ignored in the identification of any task-level action patterns.

Given that the two collaboration-level action patterns found in the master workspace of Group 9, `Group-Discussion-Pattern` and `Artefact-Exchange-Pattern`, are connected through a common object, the multi-role Member, these two action patterns were mapped to a single task-level action pattern. In this case, given that both the collaboration-level action patterns involved were performed by a single role and involved

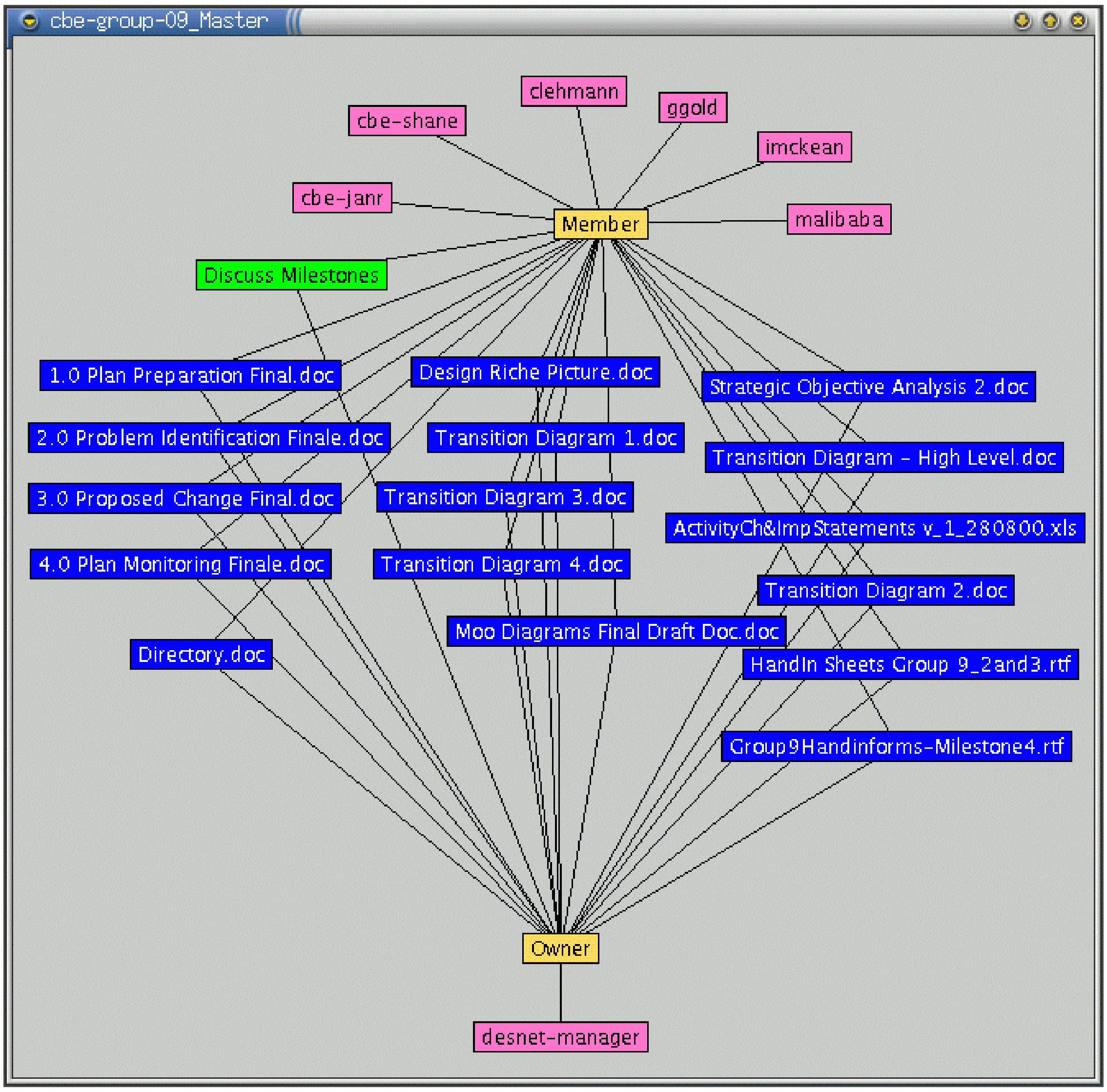

cbe-group-09_Master
clehmann
ggold
cbe-shane
imckean
cbe-janr
Member
malibaba
Discuss Milestones
1.0 Plan Preparation Final.doc
Design Riche Picture.doc
Strategic Objective Analysis 2.doc
2.0 Problem Identification Finale.doc
Transition Diagram 1.doc
Transition Diagram - High Level.doc
3.0 Proposed Change Final.doc
Transition Diagram 3.doc
ActivityCh&ImpStatements v_1_280800.xls
4.0 Plan Monitoring Finale.doc
Transition Diagram 4.doc
Transition Diagram 2.doc
Moo Diagrams Final Draft Doc.doc
Directory.doc
HandIn Sheets Group 9_2and3.rtf
Group9Handinforms-Milestone4.rtf
Owner
desnet-manager

Figure 5.16: Intra-workspace map of the master workspace of Group 9

all members of Group 9, a task-level action pattern of “collaborative report preparation” was defined as corresponding to the task performed by the group. Represented in the form of an EMOO diagram, this action pattern is shown in Figure 5.17.

5.2.2.4 Mapping Action Patterns from Task Level to Process Level

After having identified the task-level action pattern `Collaborative-Report-Preparation-Pattern`, the next step was to investigate whether this task was part of a larger process. This involved identifying information flows to or from other tasks. To aid in this, the LIVENET Workspace Visualizer was once again used. An information flow exists when two workspaces have a shared document, background, discussion, or message

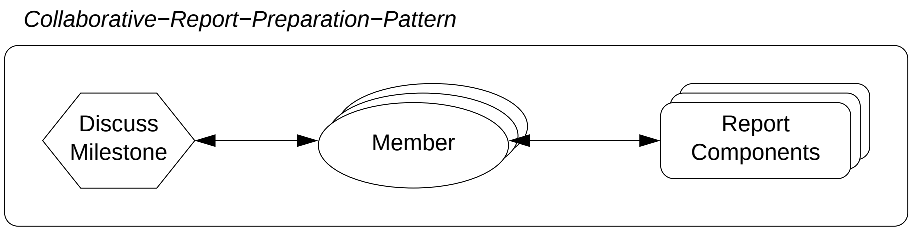


Figure 5.17: EMOO diagram representing an instance of task-level action pattern Collaborative-Report-Preparation-Pattern, involving multi-role Member, discussion forum Discuss Milestones, and multi-artefact Report Components

rule (the only way that information can flow from one workspace to another in LIVENET). An inter-workspace map involving the master workspace of Group 9, and showing all workspaces of the workgroup cbe-group-09 of which it was a part, as well as visualizing inter-workspace relationships of shared documents, backgrounds, discussions, and message rules, is shown in Figure 5.18. It can be seen that the entire workgroup only consisted of two workspaces, the master workspace and another workspace, Milestones4&5. Moreover, no shared documents, backgrounds, discussions, or message rules existed (the only link that exists between the two shown workspaces is a parent-child link). Therefore the `Collaborative-Report-Preparation-Pattern` task-level action pattern was not part of a larger process, and the mapping of action patterns stopped at this point.

5.2.3 Discussion

The pattern extraction shown above for Group 9 started with records at the base level of the Information Pyramid of the system through which the virtual collaboration was conducted—in this case the LIVENET system—and has through successive steps extracted progressively larger-scale patterns of virtual collaboration from it. In the end, a task-level action pattern of collaborative report preparation was identified, which constitutes the largest unit of activity for the given case.

Extracted patterns of virtual collaboration do not exist in isolation from one another. Rather, they are related with each other within and across levels of the Information Pyramid. Thus, given a specific instance of collaboration-level action pattern `Group-Discussion-Pattern` for example, it is possible to determine which instances of lower-level action patterns it aggregates (in this case, instances of `Post-Discussion-Statement-Pattern` and `Open-Discussion-Statement-Pattern`); on the other hand, it is also possible to say in which instances of higher-level action patterns this collaboration-level action pattern is involved (in this case, in an instance of `Collaborative-`

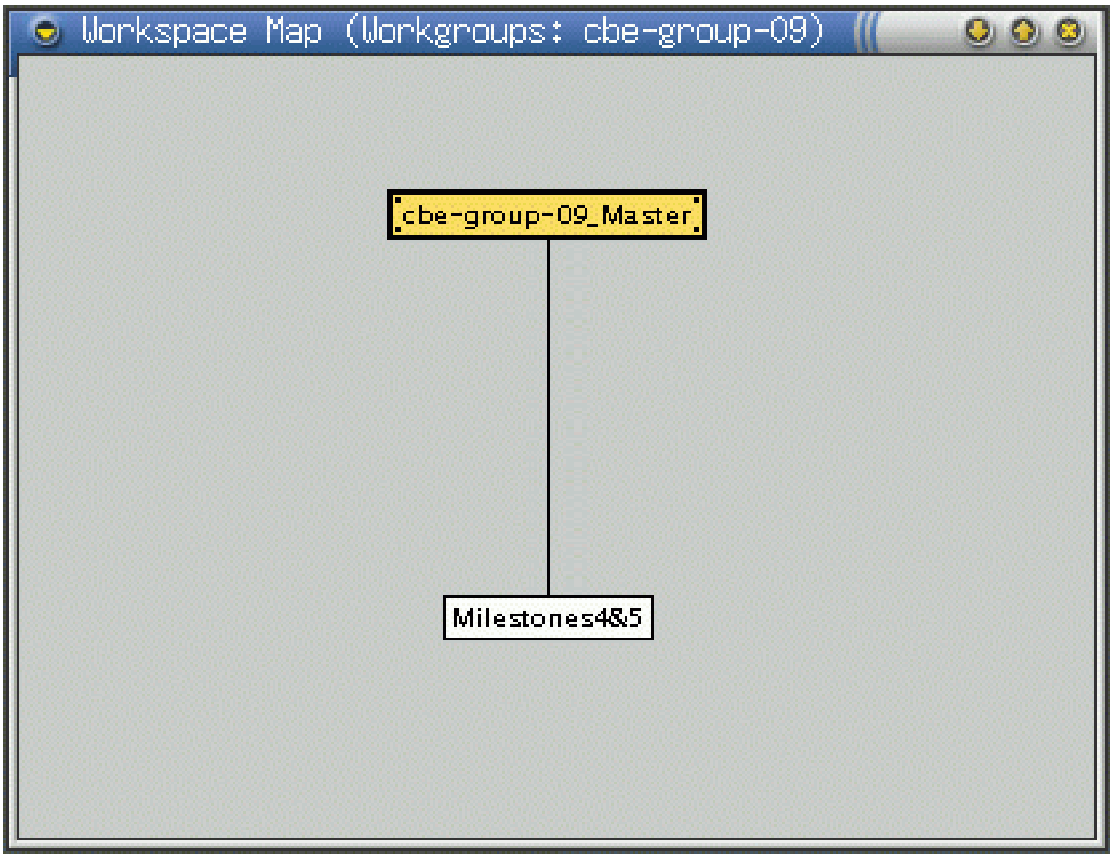


Figure 5.18: Inter-workspace map of workgroup cbe-group-09

`Report-Preparation-Pattern`). Knowing these relationships among instances of action patterns on different levels makes it possible to both abstract away details of individual users' actions in order to obtain a "big picture" of the virtual collaboration at hand; and to "drill down" into more detail of a specific action pattern whenever such information is required. Thus the result of pattern extraction is a *network* of related instances of patterns of virtual collaboration that can be traversed in any direction to obtain the information sought (on the level of instances, this corresponds to the horizontal and vertical links among concepts in the LIVENET ontology as shown in Figure 5.10 on page 190).

There is a parallel between the models and methods presented here and those of the well-known Unified Modeling Language, UML. In UML, different modeling notations exist for representing different aspects of a system, nine types of diagrams in total (Booch et al., 1999). Three of these can be regarded as similar to those used in this thesis: UML class diagrams, object diagrams, and use case diagrams. UML class and object diagrams represent details of classes and objects, respectively. These correspond more or less directly to the class and object diagrams introduced in Section 4.1.3 and used throughout this chapter. These diagrams model structural aspects of the concepts in an ontology. On the other hand, UML use case diagrams can to some extent be compared to the EMOO diagrams introduced in Section 3.3 that have also been used in this chapter. The parallel, however, extends beyond the notation only. As in the case of UML, the class, object and EMOO diagrams proposed in this thesis are mutually related: objects instantiate classes, and classes are the constituents of EMOO diagrams. Moreover, EMOO diagrams are somewhat more abstract representations, hiding the detail contained in class diagrams. The same is true in the case of UML use case diagrams which hide the detail of UML class diagrams. However, there is a major difference in the modeling process in UML and that proposed in this thesis, namely the method of deriving one type of diagram from another: in UML, first use cases are modeled and represented in use case diagrams. From these, other diagrams are then produced, including class diagrams but also other diagrams not mentioned here. These models usually begin at a high level of abstraction and subsequently become increasingly detailed as modeling progresses. The method defined in this thesis, however, is the exact reverse: the starting point in modeling is detailed data from which classes are modeled. These classes are then abstracted to EMOO diagrams, and both class and EMOO diagrams become increasingly abstract as modeling progresses. Thus while there are parallels between the modeling methods and notations of UML and this thesis, the modeling approach is fundamentally different.

Pattern extraction, such as the one illustrated above, over time results in a large number of instances of different patterns of virtual collaboration at different levels of abstraction. When fed into a collaboration memory, as proposed in Section 4.5, these provide a ready resource that can be tapped into to obtain information related to the procedural aspects of the work of virtual teams. These patterns thus form the observations of virtual collaboration referred to at the beginning of this thesis, which help address the two challenges posed in Chapter 1: how to know how to carry out collaboration virtually; and how to know what is, and has been, “going on” during virtual collaboration.

In relation to the first challenge, consulting such a collaboration memory containing patterns of virtual collaboration it is possible to learn from the experience of others that is enshrined in these patterns. For instance, before setting out on an activity of virtual collaboration, one may refer to a number of patterns originating from those whom one knows to be more experienced in collaborating virtually, thereby drawing on their experience. A collaboration system with an integrated collaboration memory component could make patterns of virtual collaboration available for instantiation, possibly involving a first step of tailoring these patterns to suit the specific needs of those who are to use them. In this regard the use of information visualization can be useful in observing initial patterns, as was demonstrated above. It can also facilitate referencing and looking up patterns from the collaboration memory. Moreover, a collaboration system could also provide facilities for combining several separate patterns ("plugging" them together, as suggested earlier in this chapter), thereby effectively using these patterns as basic building blocks for creating the collaboration support required for specific collaborative endeavours.

In relation to the second challenge, those who require information about the activities of virtual teams can similarly reference records of the virtual collaboration deposited in the form of patterns of virtual collaboration in a collaboration memory. The information within these patterns then provides insights into the procedural aspects of a virtual team's collaboration. This makes it possible to determine the kinds of collaborative activities performed, the involvement of different team members in these activities, and the development of the virtual work over time.

By using the methods proposed in the preceding chapters of this thesis, these observations of virtual collaboration are obtained without requiring virtual teams to explicitly document their own actions, thereby addressing the research problem posed in Chapter 1. The preceding case study has thus provided basic validation of the plausibility and applicability of the proposed concepts and methods.

5.3 Summary

This chapter has presented a case study demonstrating the application of the concepts and methods introduced in the preceding two chapters.

The basis for being able to extract patterns of virtual collaboration from a body of data is a clear understanding of the concepts and their relationships represented by that data, as well as the abstractions that can be derived from these concepts. These are specified in the form of an ontology for a given collaboration system. An example of the specification of such an ontology for the LIVENET collaboration system was given in the first part of this chapter. Because of space considerations, this specification was illustrated by several representative examples rather than a full specification of all details.

The ontology, as specified for LIVENET, was used in the second half of this chapter in the extraction of actual instances of patterns of virtual collaboration from real usage data obtained from the LIVENET collaboration system. This illustration showed the extraction and mapping of patterns of virtual collaboration from the system level up to the task level, resulting in a network of inter-related instances of patterns of virtual collaboration that enables both abstraction and drilling down to obtain the required level of detail of information about the virtual collaboration.

Having presented and demonstrated the concepts and methods proposed in this thesis in this and the preceding chapters, the following chapter carries on from here to provide a summary and conclusions of this thesis.

Chapter 6

Summary and Conclusions

The preceding chapters have presented and illustrated all the elements this research has been concerned with. The current chapter concludes the thesis by providing a summary of the main points, listing the major contributions of this research, outlining areas for future work, and offering concluding remarks.

6.1 Summary

This thesis began with the observation that recent changes in the organizational setting have lead to an increase in virtual collaboration, understood to consist of acts of working together without face-to-face interaction, enabled by technology. Virtual collaboration is challenging, including the challenges of knowing how to carry out collaboration virtually, and knowing what is and has been “going on” during virtual collaboration. It was suggested that observation of virtual collaboration, both past and present, can help meet these challenges, but that this brings up what constitutes the problem motivating this research: how to obtain these observations without requiring those involved in the collaboration to document their own actions.

It was then suggested that the solution to this problem involves three components: firstly, the availability of records of events transpiring during virtual collaboration; secondly, conceptual modeling of the information of the computer-based systems through which virtual collaboration is conducted; and thirdly, the derivation of abstract representations of the virtual collaboration.

After a review of the main areas of the problem domain of this research in Chapter 2, the main body of the thesis, Chapters 3 and 4, presented details of the problem solution.

The main conceptual element proposed is that of a *pattern of virtual collaboration*, which is understood to refer to a structure existing within a body of data about virtual collaboration, and which represents both a particular setup of a virtual space and the actions performed within it. Patterns of virtual collaboration can exist at different levels of abstraction, ranging from highly detailed expressions of parts of a specific user action to abstract representations of entire work processes. The thesis proposed a layered model of information, the *Information Pyramid of Virtual Collaboration* which consists of six levels covering this range. Collaboration systems constitute the source of information, providing information which typically belongs to the lowest one or two levels of the Information Pyramid. On the other hand, the observations of virtual collaboration that are sought usually reside on the highest two levels of the Information Pyramid.

In order to bridge from the information source (the input) to the observations sought (the output), information at the different levels needs to be transformed to successively higher levels in the Information Pyramid. This requires two steps of (1) modeling of information, and (2) transformation of models of information from one level to another.

Information from a given collaboration system is specified in the form of an ontology, capturing concepts and their relationships. This ontology of just one level of the Information Pyramid is extended by adding concepts corresponding to the higher levels of the Information Pyramid, and by defining mappings between concepts on adjacent levels. The resulting ontology defines how information on higher levels can be obtained from that at lower levels. The actual transformation of information across levels is performed by *mapping functions*.

To aid the identification of patterns on the highest two levels of the Information Pyramid, the use of information visualization was proposed. Networks of related objects are visualized as node-and-link diagrams, while *measures of collaboration spaces* are visualized in order to aid comparison of different collaboration spaces by certain derived properties.

A *Framework for Pattern Extraction and Feedback* was proposed, which considers the extraction of patterns of virtual collaboration in the larger context of the development and utilization of collaboration systems. The framework includes four areas: collaboration systems, collaboration data, pattern extraction, and collaboration memory. The latter, collaboration memory, was proposed as a particular kind of organizational memory, consisting of patterns of collaboration.

Finally, the presented concepts and methods of modeling and derivation of patterns of virtual collaboration were demonstrated on an actual collaboration system, LIVENET.

6.2 Contributions of this Research

Following are the main contributions of the research presented in this thesis:

1. Introduction of the concept of *patterns of virtual collaboration*, as abstractions of collaborative activities at different levels of granularity.

2. The *Information Pyramid of Virtual Collaboration*, which provides views of information related to virtual collaboration on six layers with different degrees of abstraction.

3. A *method for deriving abstract representations of virtual collaboration*, given only detailed, fine-grained usage data from collaboration systems.

4. The notion of *measures of collaboration spaces* which allow different collaboration spaces to be compared based on certain derived properties.

5. A *Framework for Pattern Extraction and Feedback* which suggests ways in which collaboration systems can provide the data needed for extraction of patterns of virtual collaboration, and how these patterns can feed back into use of these systems.

6. The use of an *ontology-based notation* for the representation of patterns of virtual collaboration.

7. The notion of *collaboration memory*, as part of an organizational memory, which contains records of procedural aspects of collaborative activity.

6.3 Future Work

The research presented in this thesis has proposed ways how to both model and derive patterns of virtual collaboration from a given collaboration system. Building on this work, a number of questions arise which future research should address:

- *Rigorous empirical testing*: The previous chapter has provided some extent of validation of the plausibility of the concepts and methods proposed in this thesis. In order to demonstrate their robustness, rigorous empirical testing should be carried out. Such testing should apply these concepts and methods to a wider range of situations, involving different collaboration systems, different types of users, and different kinds of work activities.

- *Transferability of patterns*: The proposed method of pattern extraction is capable of producing patterns of virtual collaboration from a given collaboration system. An interesting problem to investigate is whether it is possible to transfer these patterns across different collaboration systems, and what would be involved in doing so.

That is, if a pattern of virtual collaboration were obtained from System X, could that pattern be transferred and utilized in System Y, and if so, how? Making it possible to transfer patterns across different collaboration systems would vastly broaden the applicability of these patterns, and thus the opportunity to benefit from the experience of others enshrined in these patterns.

Approaching this problem, the specification of ontologies for different collaboration systems, as described in this thesis, would be a required first step toward enabling transferability of patterns. Separate ontologies would, however, need to be reconciled with each other. That is, a given concept, say "Collaboration Space", may have a certain meaning and have associated with it certain functional characteristics, which may differ from one collaboration system to another. Thus the identification, and resolution, of such conceptual incongruities would likely be one of the requirements for transferring patterns across different collaboration systems. Other obstacles to the transferability of patterns may exist and will need to be investigated.

- *Accommodation of individual/group preferences*: Patterns of virtual collaboration that are obtained from a collaboration system and deposited in collaboration memory are available for instantiation, and thus reuse, by others. However, because different individuals and/or groups may have different preferred ways of working, and thus different ways of structuring their virtual working environment, the instantiation of a given pattern may not provide the best possible support for a given user or group, and may therefore need to be manually customized.

  An approach for providing instantiations of patterns of virtual collaborations that are more congruent with the individual and group preferences could involve adaptation of patterns based on *user and group profiles*. Such profiles could be obtained through observation of the way that collaboration spaces are typically set up. The way in which patterns are typically adapted before use could provide another source of information for the creation of such profiles.

- *Evaluation of patterns of virtual collaboration*: The patterns of virtual collaboration extracted from collaboration systems and retained in collaboration memory may include a wide range of patterns for different purposes as well as preferences. However, it may be difficult to know which of these patterns are best suited for a given work activity. Some patterns may facilitate a given work activity more than others, while some patterns may even encumber work. Thus a problem that exists with regard to the patterns contained in a collaboration memory is that of evaluation of their usefulness. Its solution could require a manual approach, such as that of recommender systems (Resnick and Varian, 1997), or it could involve methods of automatically evaluating patterns. The solution to this problem should be the subject of further investigation.

6.4 Concluding Remarks

Virtual collaboration has become part of professional practice in many organizations, and it is likely for this trend to continue, and indeed to increase, for the foreseeable future. The goal of many researchers is to eventually create realistic, three-dimensional, life-size virtual teleconferencing, where life-size images of other participants are projected into one's own working environment[1]. Once such technology matures and becomes widespread, it will be possible to realistically emulate face-to-face collaboration across a distance, combining the advantages of collaboration in the physical world with those that virtual collaboration has to offer.

At the present time, however, the technological support for virtual collaboration is still very rudimentary, and carrying out virtual collaboration is challenging—as was discussed in Chapter 1. To help meet these challenges, it was suggested that observations of virtual collaboration can be utilized. Addressing the problem of how to obtain observations of the activities involved in virtual collaboration without requiring virtual teams to document their own actions, this thesis has suggested a possible solution.

Collaboration is a complex phenomenon, involving multiple actors, organizations, activities, artefacts, etc., related to one another in multiple and often complex ways. A complex phenomenon such as this may be regarded as a *system*, in the sense of systems theory (Ackoff, 1971). As defined by Ackoff, a system is a "set of inter-related elements" (*ibid.*). Based on general systems theory, systems analysis approaches common in the information systems field (Avison and Fitzgerald, 1988) facilitate individual components of systems to be examined and critically reviewed. Decomposing a system into its constituent parts facilitates both the understanding of each part, as well as increasing understanding of the whole. That is what this thesis has done: the concepts presented—patterns of virtual collaboration and the multi-layered Information Pyramid—allow virtual collaboration to be examined in terms of (at least some, if not all, of) its constituent elements. The methods for derivation of higher-level patterns assist in unearthing yet more elements of the overall collaboration. Each of these constituent elements of the

[1]Early prototypes of what may be predecessors of such systems have already been developed. For an example of *tele-immersion* see (Sadagic et al., 2001), and for one of *augmented reality conferencing* see (Billinghurst and Kato, 2002).

overall collaboration can be examined in order to understand that one element, as well as contributing to the understanding of the overall complex phenomenon which is the virtual collaboration itself. Whereas obtaining a *complete* understanding of the complexities of virtual collaboration may be difficult or even impossible, the contributions made by this thesis assist in obtaining at least some degree of understanding.

Furthermore, it is important to realize that obtaining an understanding of virtual collaboration is limited by, among other factors, the degree to which collaboration is carried out virtually. Activities of virtual collaboration are usually embedded within a larger context of work in an organization, and may indeed only constitute a small part of the overall work activities of that organization. It may be complemented by other forms of collaboration, including face-to-face encounters, and may involve technologies other than collaboration systems. Therefore, insights into virtual collaboration that are obtained through the aid of the contributions made by this thesis should be complemented by insights obtained from other sources, such as from one's physical (i.e. non-virtual) work environment.

Much further work still lies ahead, some of which has been outlined in the preceding section. It is hoped that through the contributions made by this research some progress toward overcoming the challenges of collaborating virtually, which were identified in the introduction of this thesis, has been made.

Appendix A

LiveNet Workspace Visualizer

In Chapter 4, the use of information visualization was proposed for assisting the identification of action patterns at the task and process levels of the Information Pyramid. Some basic functional requirements of a visualization tool were presented, and the *node-and-link* diagram type was identified as suitable for visualizing patterns of virtual collaboration at the task and process levels.

For visualizing task- and process-level action patterns in the LiveNet collaboration system, the **Workspace Visualizer** tool was developed[1]. This tool implements the functional requirements proposed in Chapter 4 (cf. p. 131), and employs the *node-and-link* diagram type in its visualizations.

Task- and process-level action patterns in LiveNet exist either in individual workspaces (as is the case for most task-level action patterns), or span a network of workspaces (as is the case for most process-level action patterns). Thus, the Workspace Visualizer provides two basic types of diagrams: one that displays the internal structure of a workspace, and the other that displays the relationships among a network of workspaces. The following sections introduce the visualizations of the Workspace Visualizer for each of these types of diagrams.

A.1 Intra-Workspace Maps

The most basic kind of visualization is the so-called *intra-workspace map*. This makes the internal structure of a workspace visible. The internal structure is made up of the objects contained in the workspace, and the relationships between these objects. As this

[1]This tool was developed by the present writer as an 8600+ line-of-code Java program consisting of over 40 classes and interfaces. About 5% of the code is based on (now heavily modified) code produced by Sun Microsystems, Inc.

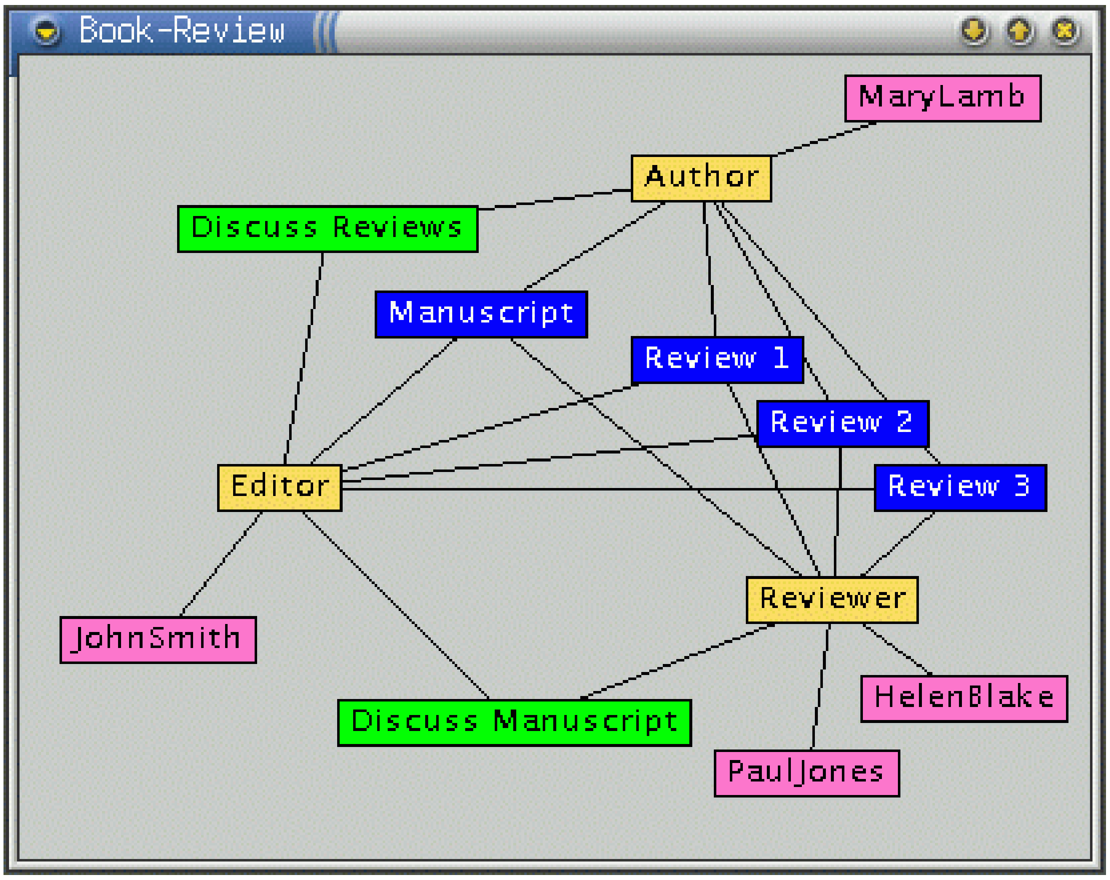


Figure A.1: Intra-workspace map of workspace `Book-Review` showing users (pink), roles (yellow), documents (blue) and discussion forums (green)

visualization focuses on the *structure* of the workspace, it draws on static information obtained from the LIVENET database. However, since the relationships between roles and other objects also imply certain actions, the visualized structure does to some extent give an indication of the behaviour, or dynamic aspects, supported by the workspace.

An example of an intra-workspace map is shown in Figure A.1. The map is a node-and-link diagram in which objects are visualized in the form of nodes, and relationships between objects are visualized as links between nodes. Different *types* of nodes are displayed with different background colours; in the example, participants (users) are shown in pink, roles in yellow, documents in blue, and discussion forums in green. Other types of objects that an intra-workspace map may contain are actions, backgrounds, message rules, and message types.

Two types of relationships are represented by links between nodes: a link connecting a participant node and a role node indicates that the participant occupies the role. A link connecting a role node and any other kind of node (action, discussion, document,

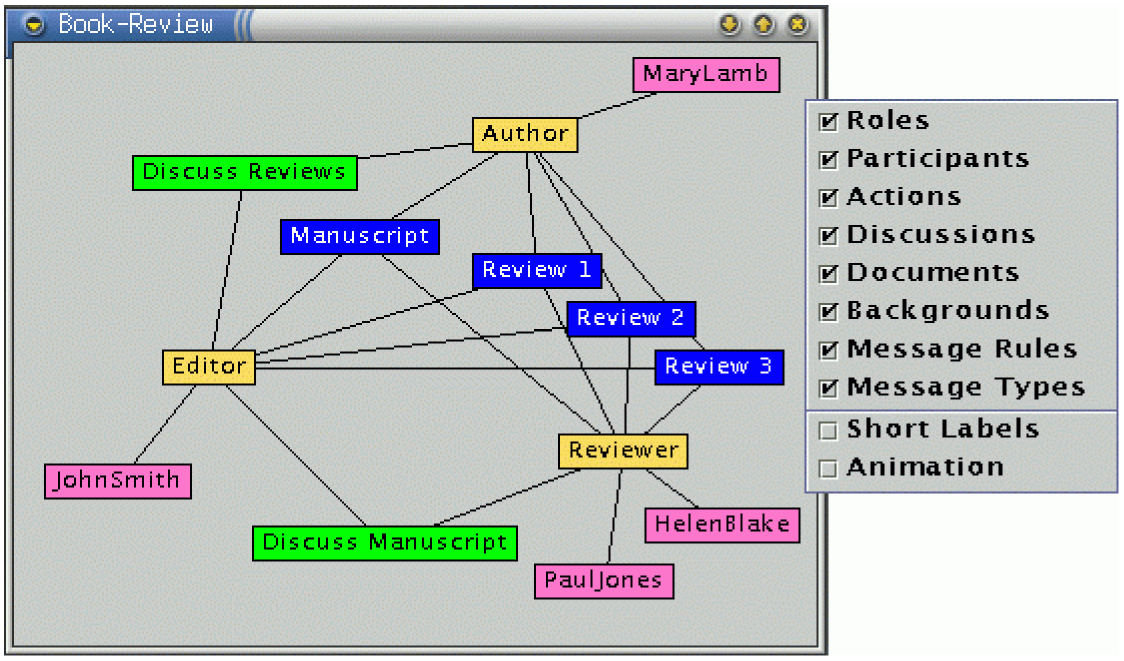


Figure A.2: Intra-workspace map with popup menu for filtering objects by type

background, message rule, or message type) indicates that the role has access to that object.

A.1.1 Filtering

For workspaces that contain many different objects, intra-workspace maps may contain a large number of nodes and links. This may make it difficult to discern the task-level action patterns contained in those workspaces. The example of Figure A.1 contains only a relatively small number of objects: four users, three roles, four documents, and two discussion forums. Yet even in this simple map there are some crossing links. For maps containing a much larger number of objects, the number of crossing links would be much greater, which can make it difficult to identify which nodes are connected by which links.

In order to facilitate the identification of patterns of virtual collaboration in workspaces, it is therefore often necessary to *filter out* some objects, i.e. to remove some objects from view, so that other objects can be more easily discerned. For the intra-workspace maps of the Workspace Visualizer, objects can be filtered out according to their *type*. This is shown in Figure A.2: the popup menu allows each of the eight types of objects listed in the upper part of the menu to be excluded from the map. By default all types of objects are included, but by unchecking the checkbox next to an object type, all objects of that type, as well as all links attached to it, are hidden.

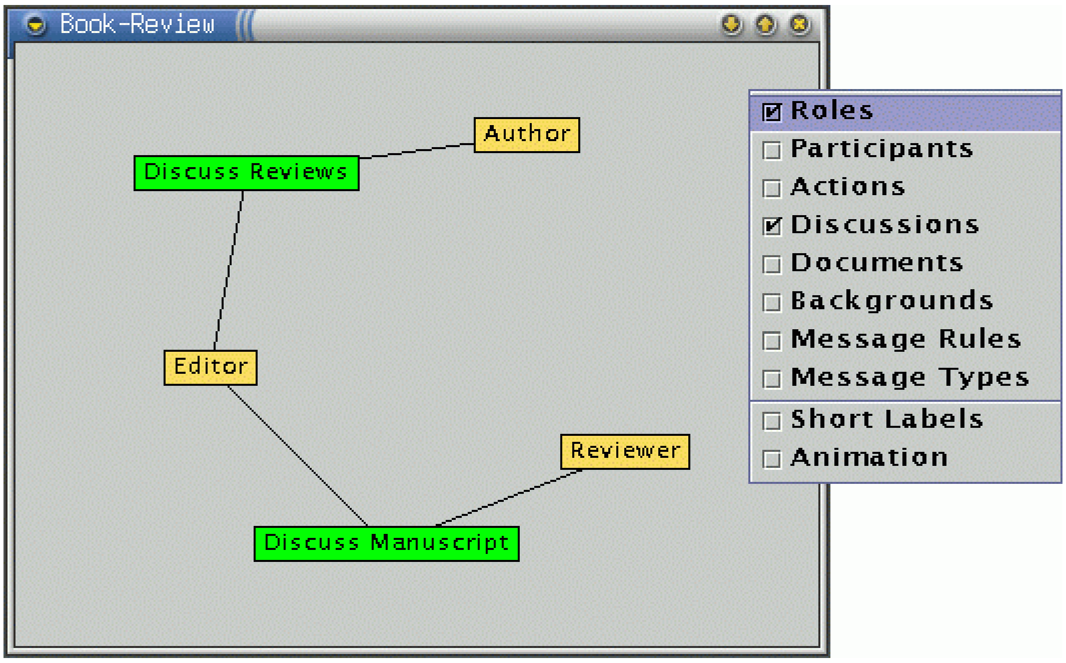


Figure A.3: Intra-workspace map with only role and discussion forum objects visible

Figure A.3 shows a visualization of the same workspace where all objects except for roles and discussion forums are filtered out. The resulting map is much clearer than the original one that contained all objects. For instance, it is now clearly visible that there are two separate instances of group discussions involving two roles each, with one of the roles present in both group discussions.

Similar filtering can be performed to leave only roles and documents visible. This allows instances of document exchanges among roles to be identified. Figure A.4 shows an example of such an intra-workspace map.

A.2 Inter-Workspace Maps

The intra-workspace maps presented above allow details of individual workspaces to be investigated. However, in order to identify process-level action patterns, it is necessary to examine not only single workspaces in isolation but groups of workspaces that are related in some way. For instance, examining a workspace such as `Book-Review` shown in Figure A.1 may lead to the identification of a task-level action pattern. Consequently other workspaces related to this one may be investigated to identify other related task-level action patterns, as well as to identify a process-level action pattern which these task-level action patterns belong to. To facilitate this, it is necessary to visualize *networks* of

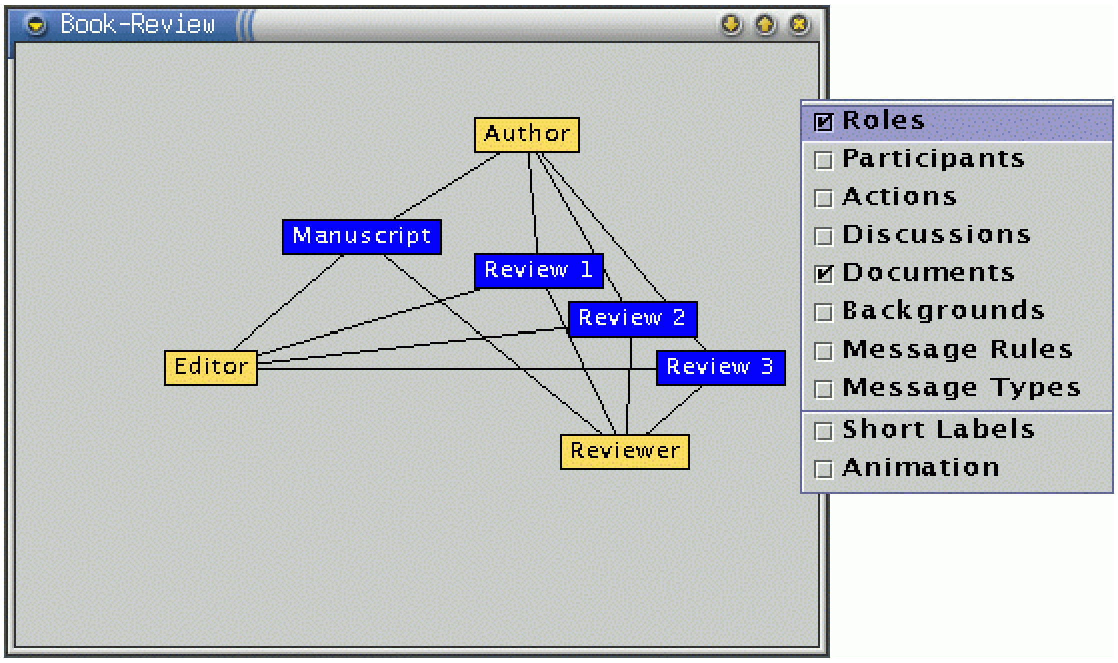


Figure A.4: Intra-workspace map with only role and document objects visible

workspaces, and to *navigate* between them. The Workspace Visualizer supports this in the form of *inter-workspace maps*.

An inter-workspace map shows several workspaces together in a node-and-link diagram, with nodes representing workspaces and links representing relationships between workspaces. Workspaces may be related in different ways. For instance, if a given document is included in two or more workspaces, these workspaces are related by participating in the sharing of the document. Other types of relationships according to shared objects can be similarly identified. Moreover, in LIVENET workspaces are related in a hierarchical structure in the form of a tree: with the exception of one workspace at the root of the tree, every workspace has a parent workspace, or conversely, workspaces may have child workspaces. Thus the visualization of workspace relationships in a collection of LIVENET workspaces consists of a tree of parent-child workspace relationships, overlaid by a network of shared-object relationships.

An example of a basic inter-workspace map, showing only the tree structure of parent-child workspace relationships, is shown in Figure A.5. Here, nodes with a yellow background are parent workspaces, i.e. workspaces which have child workspaces, while nodes with a white background are leaf nodes in the tree which have no child workspaces. The node at the top of the map is the root workspace, having no parent workspace.

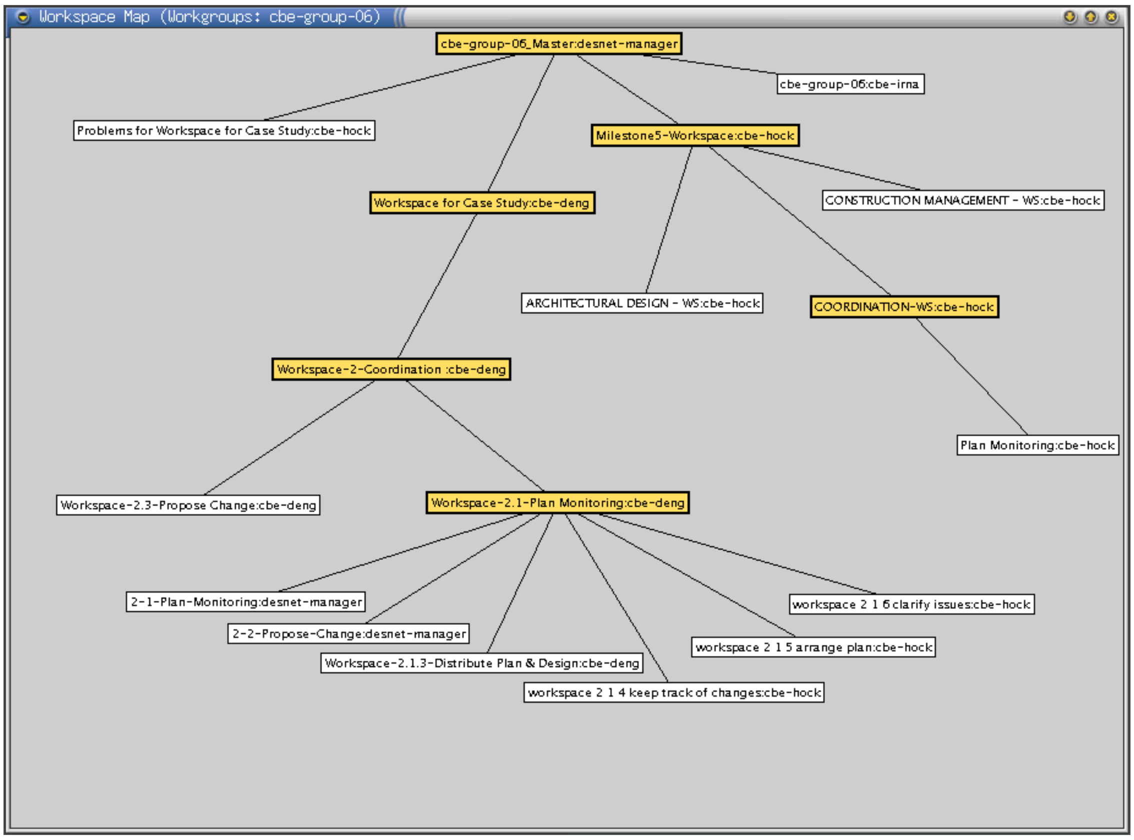

Workspace Map (Workgroups: cbe-group-06)
cbe-group-06_Master:desnet-manager
cbe-group-06:cbe-irna
Problems for Workspace for Case Study:cbe-hock
Milestone5-Workspace:cbe-hock
Workspace for Case Study:cbe-deng
CONSTRUCTION MANAGEMENT - WS:cbe-hock
ARCHITECTURAL DESIGN - WS:cbe-hock
COORDINATION-WS:cbe-hock
Workspace-2-Coordination :cbe-deng
Plan Monitoring:cbe-hock
Workspace-2.3-Propose Change:cbe-deng
Workspace-2.1-Plan Monitoring:cbe-deng
2-1-Plan-Monitoring:desnet-manager
workspace 2 1 6 clarify issues:cbe-hock
2-2-Propose-Change:desnet-manager
workspace 2 1 5 arrange plan:cbe-hock
Workspace-2.1.3-Distribute Plan & Design:cbe-deng
workspace 2 1 4 keep track of changes:cbe-hock

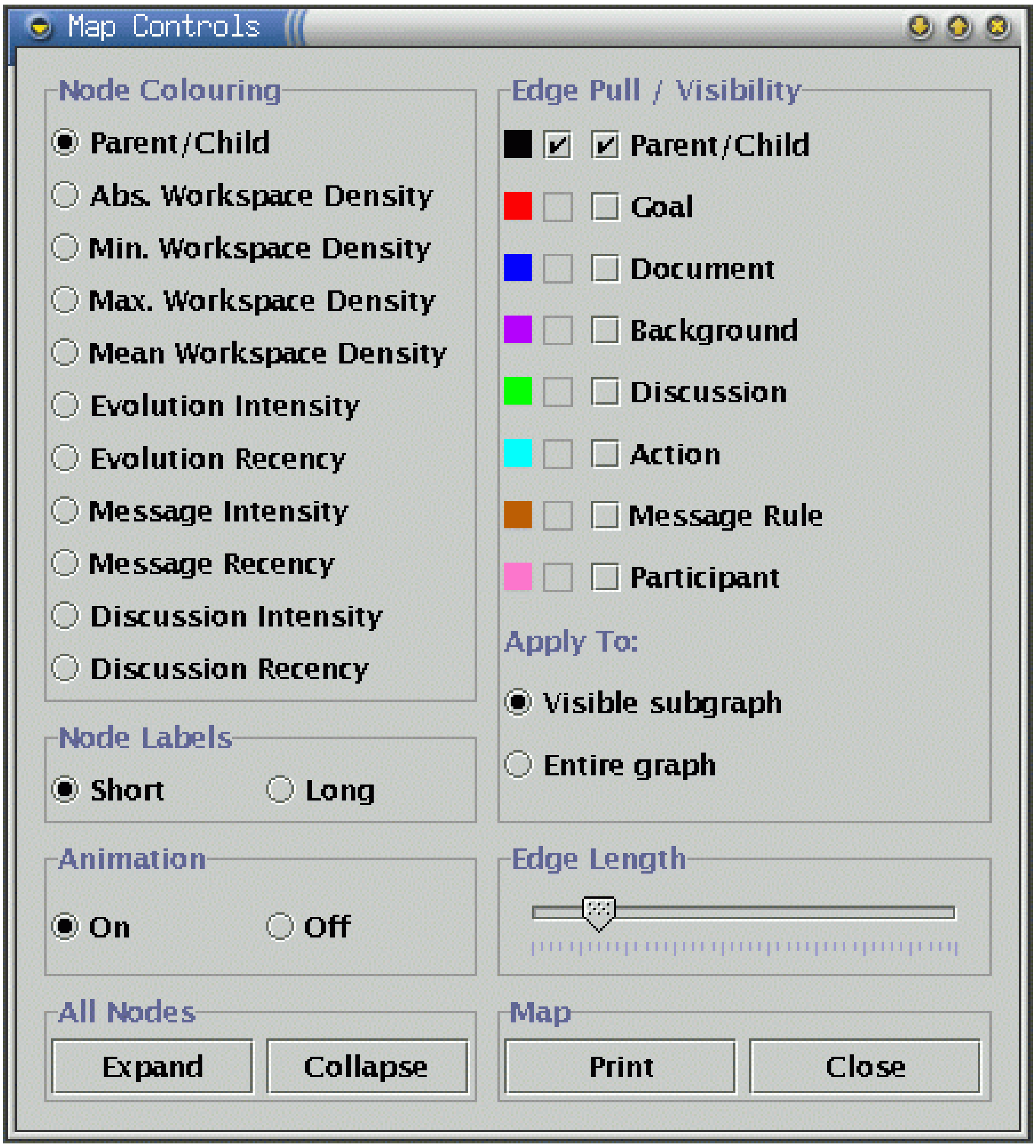
Figure A.5: Inter-workspace map of a collection of workspaces, showing parent-child workspace relationships

A.2.1 Workspace Relationship Types

Besides parent-child relationships, a number of other types of relationships can be visualized. In the Workspace Visualizer, these are represented by different link colours. Figure A.6 shows the Workspace Visualizer's control panel with which the desired workspace relationships can be selected. Here, in the section of the control panel titled "Edge Pull / Visibility", up to eight different workspace relationship types can be made visible. The colour field to the left of each relationship type label corresponds to the colour of the link shown in the map. The relationship types are:

1. **Parent/Child:** parent-child relationships between workspaces.

2. **Goal:** relationships between workspaces which support the same goal. In LIVENET, "goal" is one attribute of a workspace referring to a description of its goal. When the value of this attribute is identical for a pair of workspaces, a goal relationship exists.



Map Controls

Node Colouring

[✓ Parent/Child]
Abs. Workspace Density
Min. Workspace Density
Max. Workspace Density
Mean Workspace Density
Evolution Intensity
Evolution Recency
Message Intensity
Message Recency
Discussion Intensity
Discussion Recency

Node Labels

[✓ Short] Long

Animation

[✓ On] Off

All Nodes

Expand Collapse

Edge Pull / Visibility

⬛	✓	✓	Parent/Child
🟥			Goal
🟦			Document
🟪			Background
🟩			Discussion
🟦			Action
🟫			Message Rule
🟪			Participant

Apply To:

[✓ Visible subgraph]
Entire graph

Edge Length

Map

Print Close

Figure A.6: Control panel allowing features of the visualization to be controlled

3. **Document:** relationship between workspaces which share one or more documents.
4. **Background:** relationship between workspaces which share one or more background materials.
5. **Discussion:** relationship between workspaces which share one or more discussion forums.
6. **Action:** relationship between workspaces which share one or more actions.
7. **Message Rule:** relationship between workspaces where a message channel from one workspace to the other exists.
8. **Participant:** relationship between workspaces which have one or more participants in common.

A given collection of workspaces can potentially have a very large number of such relationships. To illustrate this, an inter-workspace map of the same collection of workspaces as shown in Figure A.5, but showing all eight types of workspace relationships, is shown in Figure A.7. As can be seen, the number of inter-workspace relationships is confusingly large. Therefore, the same principle should be applied as with the visualization of intra-workspace maps: namely to filter out information so that only the subset of information of interest is visible. To this end, three techniques are applied in the Workspace Visualizer. The first one, which filters workspace links, consists of the selection of types of relationships to be visualized, and has already been introduced. The other two, which filter workspace nodes, are *node expansion* and *focusing*, and are introduced next.

A.2.2 Node Expansion

Node expansion refers to the selective making visible of a workspace's child workspaces. Thus when the Workspace Visualizer displays an inter-workspace map, it first displays only the workspace at the root of the tree. Through a user interface action (double-clicking the mouse pointer on the workspace node), the workspace node is *expanded* to reveal its child workspaces. This can in turn be repeated on those child workspaces which are themselves parent workspaces of the next level, etc. On the other hand, the same user interface action on an already expanded workspace node *collapses* that node, i.e., hides the node's child workspaces and any lower-level workspaces for which the collapsed node is an ancestor[2]. Thus a large workspace tree can be *navigated* through

[2]An *ancestor* workspace is a workspace which is either another workspace's parent workspace, or is its parent's ancestor workspace.

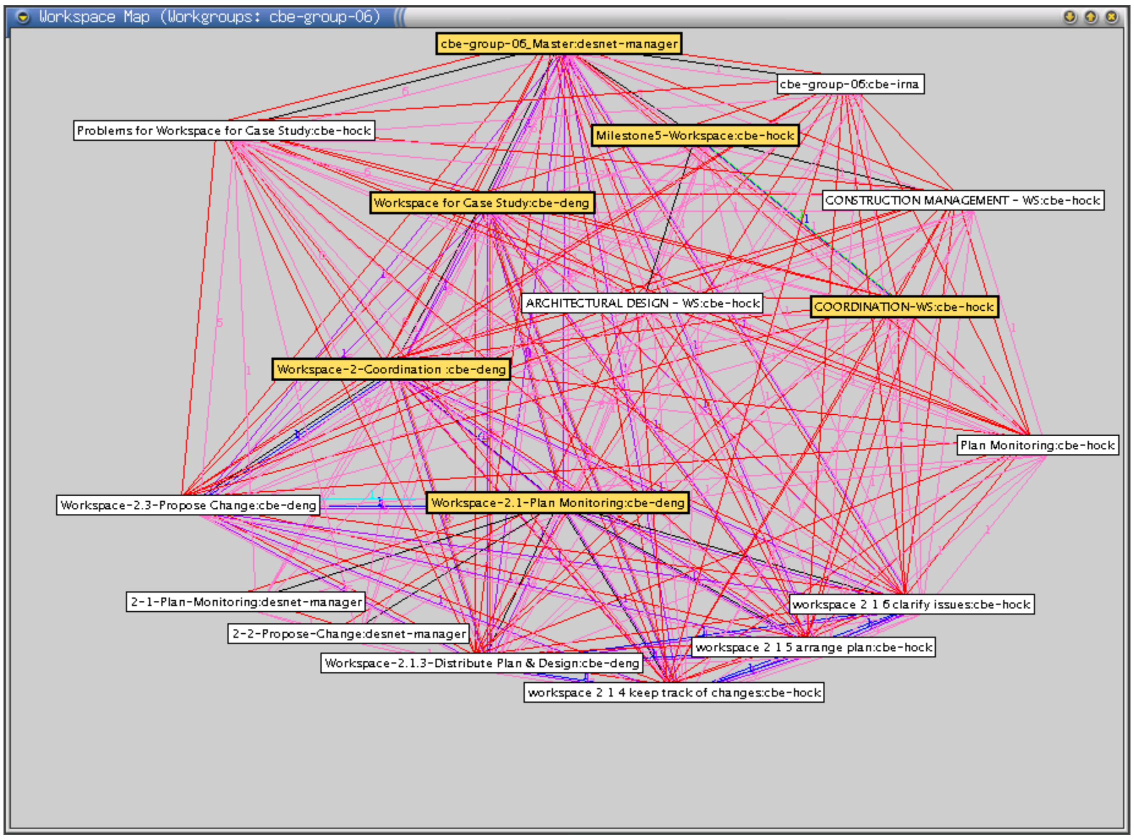

Workspace Map (Workgroups: cbe-group-06)
cbe-group-06_Master:desnet-manager
cbe-group-06:cbe-irna
Problems for Workspace for Case Study:cbe-hock
Milestone5-Workspace:cbe-hock
Workspace for Case Study:cbe-deng
CONSTRUCTION MANAGEMENT - WS:cbe-hock
ARCHITECTURAL DESIGN - WS:cbe-hock
COORDINATION-WS:cbe-hock
Workspace-2-Coordination :cbe-deng
Plan Monitoring:cbe-hock
Workspace-2.3-Propose Change:cbe-deng
Workspace-2.1-Plan Monitoring:cbe-deng
2-1-Plan-Monitoring:desnet-manager
workspace 2 1 6 clarify issues:cbe-hock
2-2-Propose-Change:desnet-manager
workspace 2 1 5 arrange plan:cbe-hock
Workspace-2.1.3-Distribute Plan & Design:cbe-deng
workspace 2 1 4 keep track of changes:cbe-hock

Figure A.7: Inter-workspace map of a collection of workspaces, showing all types of workspace relationships

a sequence of node expansion actions. An example of this is shown in Figure A.8. At first only the top-most workspace node is visible. The node is then expanded, revealing the four child workspaces shown in box 1. Next, one of these child workspaces is expanded, revealing another three workspaces, shown in box 2; lastly another one of these is expanded, revealing one more workspace, shown in box 3. In this fashion, only a relatively small subset of all workspaces in the entire graph is made visible, hiding other workspaces which are of no interest.

A.2.3 Focusing

Focusing, on the other hand, refers to a different kind of filtering of the inter-workspace map. Focusing places the focus of the visualization on a single workspace by only making that workspace and its directly related workspaces visible. The set of directly related workspaces depends on the selected types of workspace relationships. For instance, Figure A.9 shows examples of focusing on a workspace node. Part (a) of the figure shows

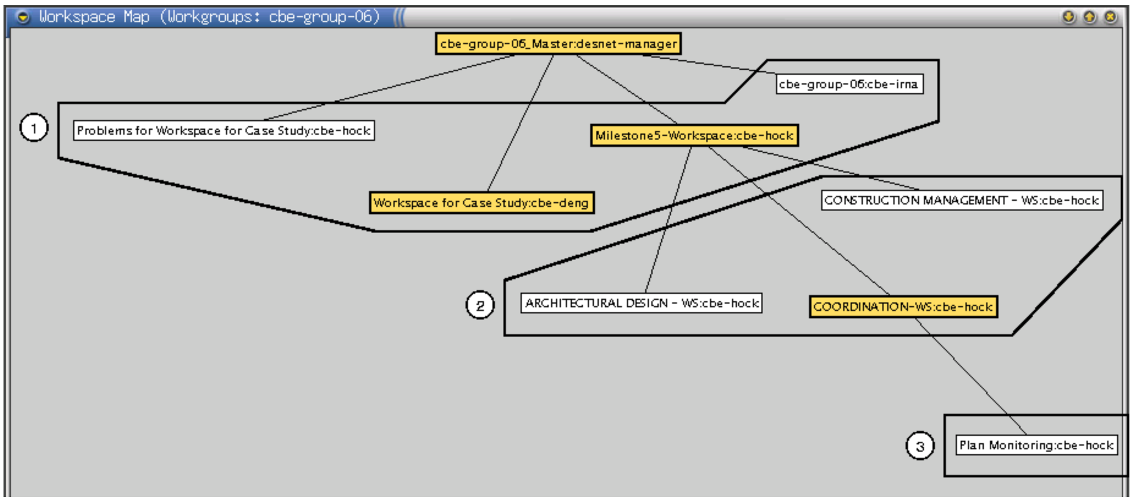
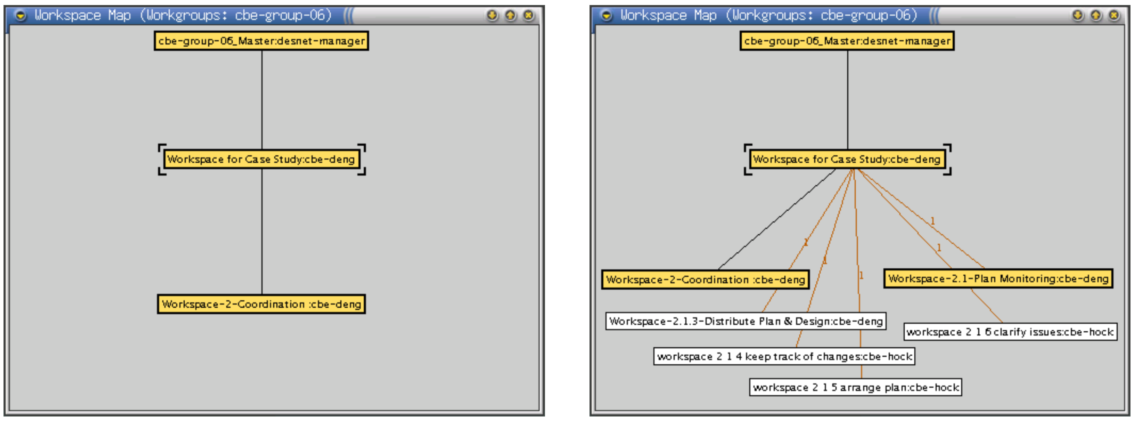

Workspace Map (Workgroups: cbe-group-06)
cbe-group-06_Master:desnet-manager
cbe-group-06:cbe-irna
Problems for Workspace for Case Study:cbe-hock
Milestone5-Workspace:cbe-hock
Workspace for Case Study:cbe-deng
CONSTRUCTION MANAGEMENT - WS:cbe-hock
ARCHITECTURAL DESIGN - WS:cbe-hock
COORDINATION-WS:cbe-hock
Plan Monitoring:cbe-hock
1
2
3

Figure A.8: Navigation of a workspace tree through a sequence of node expansion actions; workspace nodes appear in the inter-workspace map after expansion of their parent workspace, in the shown sequence (1, 2, 3).


Workspace Map (Workgroups: cbe-group-06)
cbe-group-06_Master:desnet-manager
Workspace for Case Study:cbe-deng
Workspace-2-Coordination :cbe-deng
Workspace Map (Workgroups: cbe-group-06)
cbe-group-06_Master:desnet-manager
Workspace for Case Study:cbe-deng
Workspace-2-Coordination :cbe-deng
Workspace-2.1-Plan Monitoring:cbe-deng
Workspace-2.1.3-Distribute Plan & Design:cbe-deng
workspace 2 1 6 clarify issues:cbe-hock
workspace 2 1 4 keep track of changes:cbe-hock
workspace 2 1 5 arrange plan:cbe-hock
1
1
1
1
1

(a) “Visible subgraph” link visibility

(b) “Entire graph” link visibility

Figure A.9: Focused workspace with parent-child and message rule relationships

the workspace at the centre being focused on. This is indicated by the brackets around the node's corners. Only that workspace, its parent, and its child workspace are visible in the map. Part (b) of the figure shows the same workspace being focused on, but this time includes several more related workspaces. In both cases, the workspace relationships to visualize include the parent-child and the message rule types. However, the two maps differ in the *scope* of the visibility of workspace relationships. The scope can be controlled in the lower part of the control panel's "Edge Pull / Visibility" section; it can be limited to just the currently visible subgraph, or can be set to apply to the entire graph. For instance, if a given subset of the graph such as the one in Figure A.9 (a) is currently visible, and the Message Rule relationship type is being made visible in the "Apply to visible subgraph" scope, the corresponding links will only be shown if they connect already visible workspace nodes. In this way, relationships among a set of currently displayed workspaces can be explored by switching the visibility of different link types on and off. On the other hand, if the Message Rule relationship type is being made visible in the "Apply to entire graph" scope, the corresponding links will be shown for all workspace nodes which are connected to the already visible ones, including those workspace nodes which were not previously visible. This is useful for identifying all related workspaces for a given set of workspaces. This is the case in Figure A.9 (b) where five more workspaces are now visible. These workspaces are related to the focused-on workspace through a message rule link, but are not among its parent or child workspaces.

A.2.4 Clustering

A large inter-workspace map containing dozens of workspaces may contain hundreds of workspace links. In order to identify sets of workspaces which are closely related and may belong to the same process, it can be useful to arrange these in close proximity to each other in the map. Doing so is referred to as *clustering* of workspace nodes.

To understand how clustering works, it is necessary to understand how the maps are drawn. Inter-workspace maps are drawn using a force-directed animated visualization algorithm, similar to that of (Huang et al., 1998). With this algorithm, nodes and links in the graph behave similar to round magnets connected by springs: the magnets can be imagined to have the same magnetic pole on the outside so that they repel each other, while the force of the springs pulls the magnets to a position where the springs are at their "natural" length (i.e. their length when at rest). This is illustrated in Figure A.10, which corresponds to an inter-workspace map with one root node (the circle at the top), and three child nodes (the other three circles). The four outward-pointing arrows correspond to the repelling forces of the magnets, while the arrows next to the springs correspond to

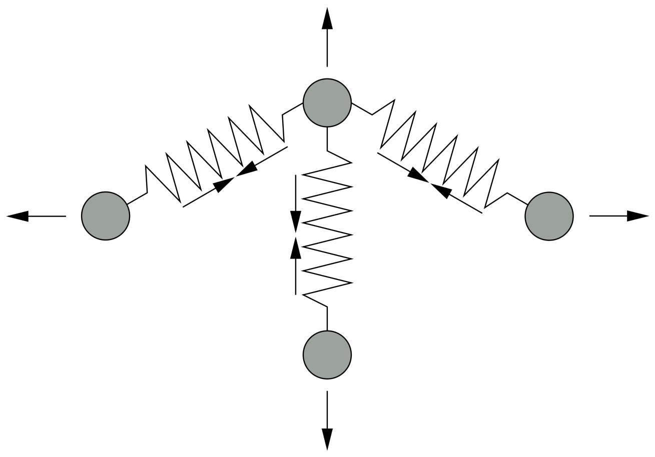
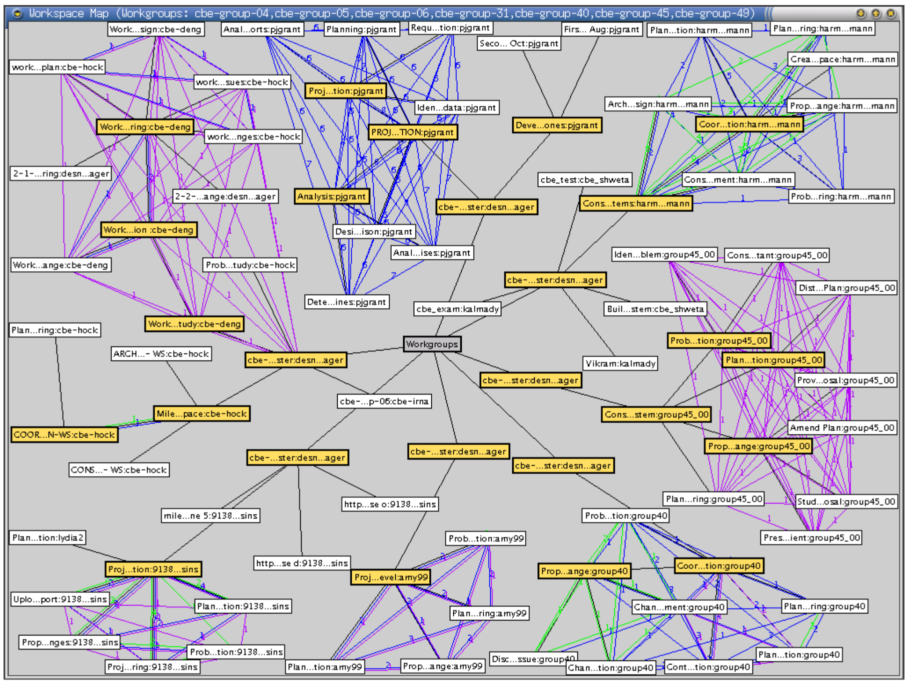


Figure A.10: Magnets-and-connecting-springs model on which the force-directed animated visualization algorithm is based

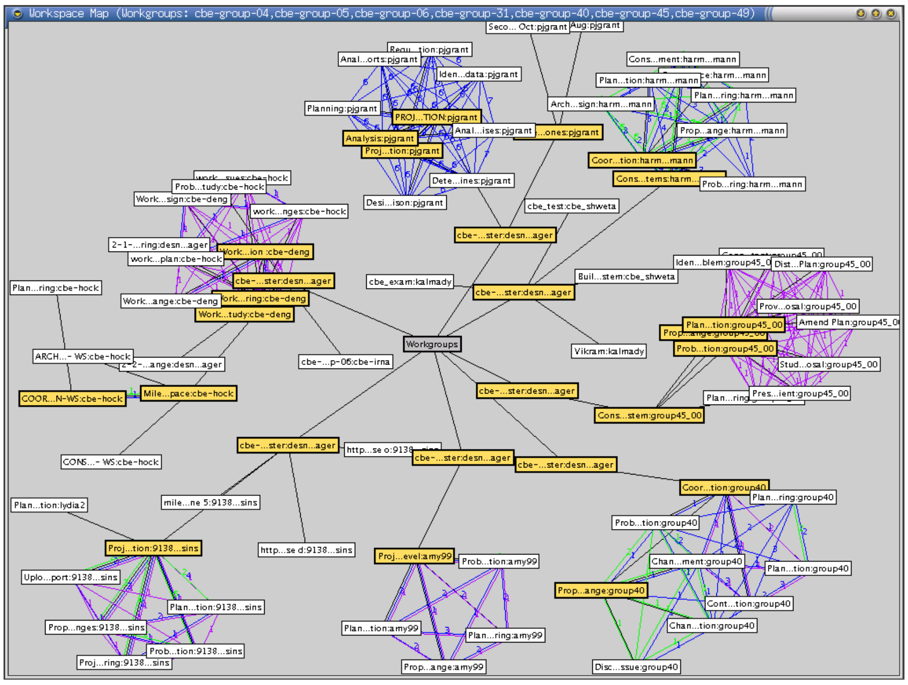
the spring forces which contract the springs to their natural length, and which counter-act the magnets' repelling forces. Other forces such as gravity are assumed to be absent. When the forces are in equilibrium, the magnets and springs are motionless. However, when one node is moved, thereby extending or contracting any connecting springs, other springs and nodes will also move until equilibrium is once again established. This effect is modeled in the visualization algorithm, where nodes repel each other, while links contract to their natural length. The natural length of links is equal for all links and can be set in the control panel's "Edge Length" section.

An extension of the algorithm of (Huang et al., 1998) in the Workspace Visualizer is the introduction of the notion of springs whose spring force can be disabled. The effect is that such a spring behaves like an infinitely stretchable string: when a magnet is attached to it and is repelled from other nearby magnets, the spring is simply extended as far as necessary. In the Workspace Visualizer, this feature is utilized in selectively enabling and disabling the spring force for different types of workspace relationship links (this feature is enabled by checking the "Edge Pull" checkbox next to a type of relationship link in the control panel). Thus, for instance, all shared-document links can be made springs that have a spring force, while the spring force of other links is disabled. The effect is that all pairs of workspace nodes which have shared documents are pulled together into close proximity (the distance being approximately that of the natural length of the link). In a large network of workspaces, this results in *clusters* of related workspaces to emerge.

An example of clustering is presented in Figure A.11. Part (a) of the figure shows an inter-workspace map with the spring force enabled only on parent-child links (the default setting). Several groups of workspaces can be identified that are related through



(a) Spring force only on parent-child links



(b) Spring force on all links

Figure A.11: Clustering of workspace nodes in an inter-workspace map with parent-child, shared document, shared background, and shared discussion forum links

shared objects. The extent to which workspaces within these groups are related, however, is not clearly discernible. Part (b) of the figure shows the same inter-workspace map but with the spring force enabled on all links, including the shared object links. The spring force has caused the groups of workspaces to contract into tight clusters. This is particularly evident for the four larger clusters in the upper part of the map. The shape of these clusters, close to that of regular polygons, indicates that the workspaces in those clusters share objects with every other workspace in the same cluster, for otherwise the combination of spring force and repulsion force would drag the cluster apart into a less regular shape. In contrast, the cluster in the lower right corner indicates just such a case. Here there appear to be two inter-woven sub-clusters: one involving shared documents and discussion forums (the lower left part of the cluster), and the other involving shared documents and background material (the upper right part of the cluster). Thus clustering can reveal not only which workspaces are related to one another, but also indicate the regularity (or lack thereof) of the network of relationships within the cluster.

A.2.5 Workspace Measures

In Chapter 4, the notion of *measures* of collaboration spaces was proposed, and a list of several such measures was presented as an illustration of this notion (cf. p. 129). In the Workspace Visualizer, the majority of these measures have been implemented, and are referred to as *workspace measures*. It should be stressed that the choice of this particular set of measures has been largely arbitrary, based purely on the observation that these measures have proven useful in distinguishing between different workspaces, but without claiming that these measures are in any way the "best" ones for comparing workspaces.

Workspace measures can be visualized in inter-workspace maps. This is done by selecting an option from the control panel's "Node Colouring" section. The default setting is "Parent/Child" colouring, which corresponds to no workspace measure being visualized, and where parent nodes appear in yellow and leaf nodes appear in white. The remaining ten node colouring options correspond to workspace measures. Each measure is mapped to a colour scale which gradually extends from one colour to another, for example from green through yellow to red. Low values of the measure are given a colour at one end of the scale, while high values receive a colour at the opposite end. The colour corresponding to the value of the measure is used as the background colour of the workspace node, providing a simple and easily perceivable distinction among a collection of workspaces. An example of node colouring in the Workspace Visualizer is shown in Figure A.12. Here, the workspace measure being represented by the node's background colour is workspace density. Green corresponds to low-density workspaces,

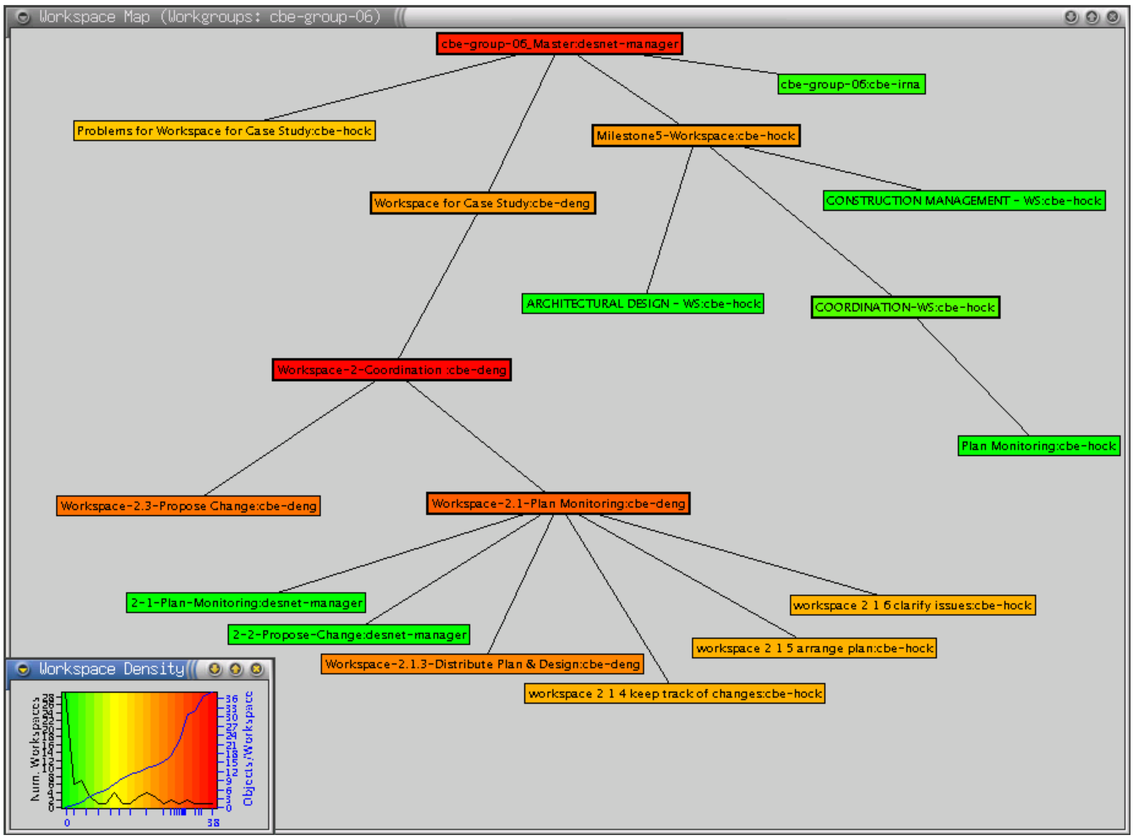

Workspace Map (Workgroups: cbe-group-06)
cbe-group-06_Master:desnet-manager
cbe-group-06:cbe-irna
Problems for Workspace for Case Study:cbe-hock
Milestone5-Workspace:cbe-hock
Workspace for Case Study:cbe-deng
CONSTRUCTION MANAGEMENT - WS:cbe-hock
ARCHITECTURAL DESIGN - WS:cbe-hock
COORDINATION-WS:cbe-hock
Workspace-2-Coordination :cbe-deng
Plan Monitoring:cbe-hock
Workspace-2.3-Propose Change:cbe-deng
Workspace-2.1-Plan Monitoring:cbe-deng
2-1-Plan-Monitoring:desnet-manager
workspace 2 1 6 clarify issues:cbe-hock
2-2-Propose-Change:desnet-manager
workspace 2 1 5 arrange plan:cbe-hock
Workspace-2.1.3-Distribute Plan & Design:cbe-deng
workspace 2 1 4 keep track of changes:cbe-hock
Workspace Density
Num. Workspaces
Objects/Workspace
0
38

Figure A.12: Inter-workspace map of a collection of workspaces, with node colouring indicating workspace density

while red correspond to those with a high density, with yellow ones lying in between. The legend in the lower left corner of the map indicates the value of the measure associated with each colour. The scale extends from a value of zero (pure green) to 38 (pure red). It can be readily seen that there are significant differences in workspace density among the 18 workspace nodes pictured: a few are deep red, i.e. very dense, while a number of workspace nodes are bright green, i.e. have low density. In deciding which workspaces to examine, one may for instance choose one each with low, average, and high density, and then compare their internal structure.

For the workspace density measure, four different options are available in the control panel: absolute, minimum, maximum, and mean workspace density. The inter-workspace map in Figure A.12 displays absolute workspace density. This is an absolute measure of the number of objects in the workspace. The other three types of workspace density measures are calculated in terms of the *perceived* density, as seen by the different roles in the workspace. Because a different subset of objects can be visible to different roles, each role may perceive a different workspace density. The minimum density is the smallest density value of all roles' perceived density; similarly, the maximum density is the highest value; and the mean density is the average of all roles' perceived density values.

The evolution intensity, message intensity, and discussion intensity values are calculated as the number of object additions, message postings, and discussion statement postings per unit of time, respectively. Finally, the evolution recency, message recency, and discussion recency values are calculated as the number of object additions, message postings, and discussion statement postings in a defined recency period, respectively. In the calculation of recency, more recent actions are given a greater weight than those further back in the past. A *deflation factor* is applied in the calculation to decrease the count given for each action the further back in the past it lies. Thus workspaces in which activity has been most recent are displayed in a colour representing a higher value than those where activity has been further back.

The ten workspace measures selectable in the Workspace Visualizer's control panel are calculated as follows:

A.2.5.1 Density measures:

There are four different workspace density measures. The basic one is **absolute workspace density**, D_{abs_w} for workspace w, which is:

$$D_{abs_w} = |\mathcal{O}_w|$$

where $\mathcal{O}_w$ is the set of objects of type action, background, document, and discussion in workspace w, and $|\mathcal{O}_w|$ is the cardinality of $\mathcal{O}_w$.

For the other types of workspace densities, it is necessary to calculate *role densities* first. The role density D_{r_w} of a given role r in workspace w is:

$$D_{r_w} = |\mathcal{O}_{wr}|$$

where $\mathcal{O}_{wr}$ is the set of objects of type action, background, document, and discussion in workspace w which are visible to role r, and where $|\mathcal{O}_{wr}|$ is the cardinality of this set.

For a given workspace w, let the set of all roles in w be denoted as $\mathcal{R}_w$, and the set of all role densities D_{r_w} in w be denoted as $\mathcal{D}_{\mathcal{R}_w}$. The remaining three types of workspace density measures are then:

Minimum workspace density, D_{min}:

$$D_{min} = \min(\mathcal{D}_{\mathcal{R}_w})$$

where the function min obtains the smallest value of D_{r_w} from $\mathcal{D}_{\mathcal{R}_w}$.

Maximum workspace density, D_{max}:

$$D_{max} = \max(\mathcal{D}_{\mathcal{R}_w})$$

where the function max obtains the largest value of D_{r_w} from $\mathcal{D}_{\mathcal{R}_w}$.

Mean workspace density, D_{mean}:

$$D_{mean} = \langle \mathcal{D}_{\mathcal{R}_w} \rangle$$

where $\langle \mathcal{D}_{\mathcal{R}_w} \rangle$ denotes the arithmetic mean of the values in $\mathcal{D}_{\mathcal{R}_w}$.

A.2.5.2 Intensity measures:

Intensity measures express the temporal clustering of a given set of action types in a workspace. There are three intensity measures in the Workspace Visualizer:

Evolution intensity, I_{evol_w}, measures the number of object additions to workspace w since the time of its creation, over the age a of the workspace's existence (in weeks):

$$I_{evol_w} = \frac{|\mathcal{O}_w|}{a}$$

where $|\mathcal{O}_w|$ is the cardinality of the set of objects added to workspace w. Objects included in $\mathcal{O}$ are: roles, participants, documents, backgrounds, discussions, actions, message types, and message rules.

Message intensity, I_{msg_w}, measures the number of message postings through message channels in workspace w since the time of its creation, over the age a of the workspace's existence (in weeks):

$$I_{msg_w} = \frac{|\mathcal{M}_w|}{a}$$

where $|\mathcal{M}_w|$ is the cardinality of the set of messages $\mathcal{M}$ posted in workspace w.

Discussion intensity, I_{disc_w}, measures the number of discussion statement postings through discussion forums in workspace w since the time of its creation, over the age a of the workspace's existence (in weeks):

$$I_{disc_w} = \frac{|\mathcal{S}_w|}{a}$$

where $|\mathcal{S}_w|$ is the cardinality of the set of discussion statements $\mathcal{S}$ posted in workspace w.

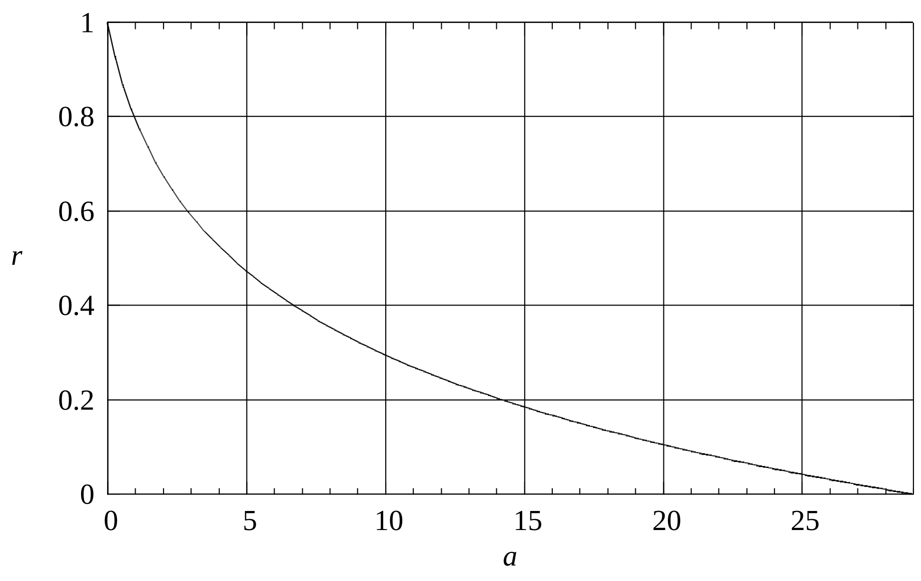


Figure A.13: Weighted action count r for recency measures, with recency period $P = 30$

A.2.5.3 Recency measures:

Recency measures express the temporal clustering of a given set of action types in a workspace over a time period ending at the time of observation. In the calculation of recency, youngest actions are given the greatest weight, which decreases with the action's age logarithmically from 1 (actions occurring on day of observation) to 0 (actions occurring outside the recency period). In the Workspace Visualizer, the recency period P is 30 days, including the day of observation and 29 days into the past. The weighted action count r for this recency period is shown in Figure A.13. There are three recency measures in the Workspace Visualizer:

Evolution recency, R_{evol_w}, measures the weighted number of object additions to workspace w during recency period P. For a workspace in which n object additions occurred within the recency period P, it is calculated as:

$$R_{evol_w} = \sum_{i=1}^{n} r_i$$

where each value of r corresponds to one object addition with age a, $a \geq 0$, in days before the time of observation, and is calculated as:

$$r_i = 1 - \frac{\ln(a+1)}{\ln(P)} \tag{A.1}$$

Message recency, R_{msg_w}, measures the weighted number of message postings through message channels in workspace w during recency period P. For a workspace in which m message postings occurred within the recency period P, it is calculated as:

$$R_{msg_w} = \sum_{i=1}^{m} r_i$$

where each value of r corresponds to one message posting with age a, $a \geq 0$, in days before the time of observation, and is again calculated as in eq. A.1 above.

Discussion recency, R_{disc_w}, measures the weighted number of discussion statement postings through message channels in workspace w during recency period P. For a workspace in which l discussion statement postings occurred within the recency period P, it is calculated as:

$$R_{disc_w} = \sum_{i=1}^{l} r_i$$

where each value of r corresponds to one discussion statement posting with age a, $a \geq 0$, in days before the time of observation, and is again calculated as in eq. A.1 above.

A.3 From Inter- to Intra-workspace Maps

Through exploration of workspace maps, both of the inter-workspace and intra-workspace type, structures of task- and process-level action patterns existing within LIVENET's collaboration data can be identified. This exploration is aided by various techniques that were presented above, including a combination of filtering and clustering techniques, as well as the visualization of workspace measures.

The process of visualization itself may be more or less targeted. In some cases, an existing workspace is known to be of interest, in which case an intra-workspace map of just that workspace can be produced for further inspection. At other times, the process is more exploratory where the visualization can aid in the identification of workspaces which appear of interest for further investigation. Such workspaces can then be examined in more detail in a subsequent step by requesting intra-workspace maps for each one. Thus the exploration usually involves both inter- and intra-workspace maps, and often alternates between the two.

Using information visualization as provided by the Workspace Visualizer, the extraction of task- and process-level action patterns from workspaces can be greatly facilitated, as compared to the direct analysis of large quantities of lower-level action patterns. An example of the application of the Workspace Visualizer to actual collaboration data obtained from the LIVENET collaboration system was presented in Section 5.2.

Appendix B

Pattern Extraction Queries

This appendix shows the queries that were performed as part of the extraction of patterns from LIVENET data, as well as the results of those queries. The source data was generated through usage of the LIVENET collaboration system by a group of LIVENET users, designated as “Group 9” (refer to the discussion of pattern extraction in Chapter 5, beginning on page 191).

B.1 Database Structure

The queries that follow are issued against the relational database table `Log` which contains records of the LIVENET action log. This table has the structure shown in Table B.1.

B.2 Sessions

Group 9 performed 103 sessions with a total of 2737 actions per session, an average of about 26 actions per session.

B.2.1 Number of Actions Per Session

The following SQL query obtained a list of session identifiers and corresponding number of actions per session:

```
SELECT SessionId, COUNT(*) AS Num
FROM   Log
WHERE  SessionId IN (
  SELECT DISTINCT SessionId
  FROM   Log
```

Column	**Description**
`LogId`	Unique identifier of each log record
`LogTimestamp`	Timestamp of the log record (date/time)
`SessionId`	Unique identifier of the session which the action recorded in the log record belongs to
`Workgroup`	Workgroup in which the action occurred
`Workspace`	Workspace in which the action occurred
`Owner`	Owner of the workspace in which the action occurred
`UserName`	Name of the subject (user) performing the action
`RoleName`	Role occupied by the user performing the action
`Action`	The action performed
`Attrib1`	Action attribute number 1
`Attrib2`	Action attribute number 2
`Attrib3`	Action attribute number 3
`Attrib4`	Action attribute number 4
`Attrib5`	Action attribute number 5
`Attrib6`	Action attribute number 6
`Attrib7`	Action attribute number 7
`Attrib8`	Action attribute number 8
`Attrib9`	Action attribute number 9
`Attrib10`	Action attribute number 10

Table B.1: Structure of database table `Log`

```
  WHERE  Workspace = 'cbe-group-09_Master'
  AND    NOT UserName = 'desnet-manager'
)
GROUP  BY SessionId
ORDER  BY SessionId
```

The nested query is necessary because some actions, such as `setworkspace`, take place before the session has entered a workspace. Using the subquery, all session ids are identified, which then allows all actions belonging to those sessions to be counted. The condition to exclude actions performed by `desnet-manager` (the course instructor and owner of the workspace) ensures that only actions performed by members of Group 9 are considered. The graph in Figure B.1 summarizes the query result. The following output is produced by the query:

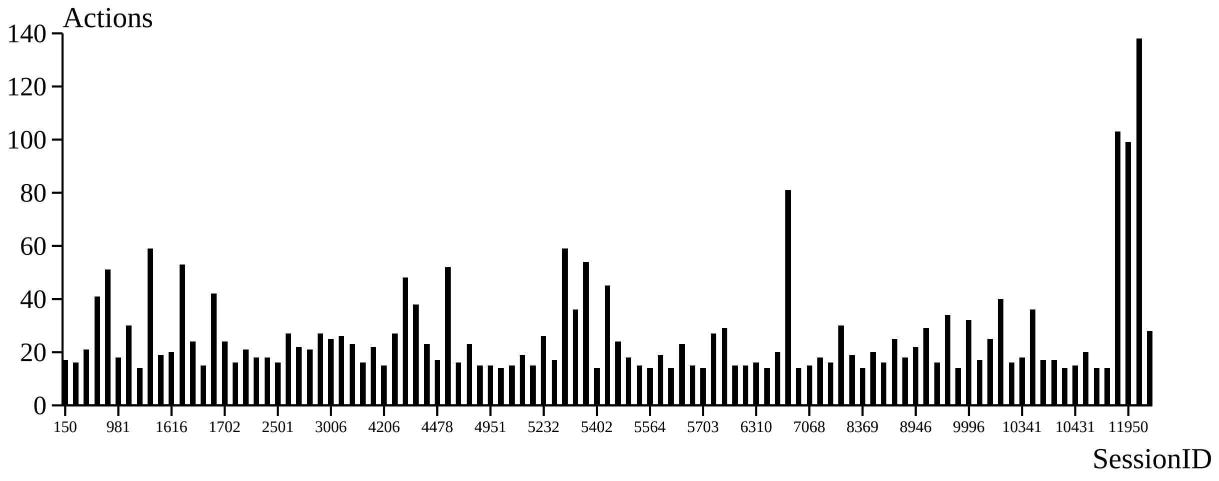


Figure B.1: Number of actions per session for Group 9

```
SessionId        Num
--------------------
  150             17
  329             16
  527             21
  673             41
  784             51
  981             18
 1198             30
 1203             14
 1209             59
 1316             19
 1616             20
 1627             53
 1643             24
 1648             15
 1701             42
 1702             24
 2039             16
 2049             21
 2058             18
 2362             18
 2501             16
 2571             27
```

```
2732                22
2742                21
2862                27
3006                25
3190                26
3250                23
3678                16
3960                22
4206                15
4207                27
4212                48
4228                38
4457                23
4478                17
4729                52
4770                16
4784                23
4808                15
4951                15
4979                14
4987                15
5122                19
5129                15
5232                26
5275                17
5281                59
5363                36
5390                54
5402                14
5404                45
5462                24
5539                18
5548                15
5564                14
5566                19
5580                14
5610                23
```

```
 5613              15
 5703              14
 5764              27
 5768              29
 5853              15
 5874              15
 6310              16
 6312              14
 6514              20
 6527              81
 6847              14
 7068              15
 7165              18
 7394              16
 7919              30
 7955              19
 8369              14
 8752              20
 8759              16
 8815              25
 8870              18
 8946              22
 9218              29
 9640              16
 9646              34
 9987              14
 9996              32
10015              17
10022              25
10039              40
10326              16
10341              18
10365              36
10374              17
10383              17
10416              14
10431              15
```

```
10580           20
11370           14
11415           14
11931          103
11950           99
11952          138
20236           28
```

B.2.2 Sessions Per Day

The following SQL query obtained a list of the number of sessions of Group 9 per day during the observation period:

```
SELECT CONVERT(CHAR(11), LogTimestamp) AS SessionDate,
       COUNT(DISTINCT SessionId) AS Num
FROM   Log
WHERE  SessionId IN (
  SELECT DISTINCT SessionId
  FROM   Log
  WHERE  Workspace = 'cbe-group-09_Master'
  AND    NOT UserName = 'desnet-manager'
)
GROUP  BY CONVERT(CHAR(11), LogTimestamp)
```

As above, the nested query obtains all session identifiers for sessions performed by members of Group 9. The graph in Figure B.2 summarizes the query result. The following output is produced by the query:

```
SessionDate           Num
-------------------------
Aug 11 2000             1
Aug 12 2000             1
Aug 13 2000             1
Aug 14 2000             1
Aug 15 2000             1
Aug 16 2000             4
Aug 17 2000             6
Aug 19 2000             3
```

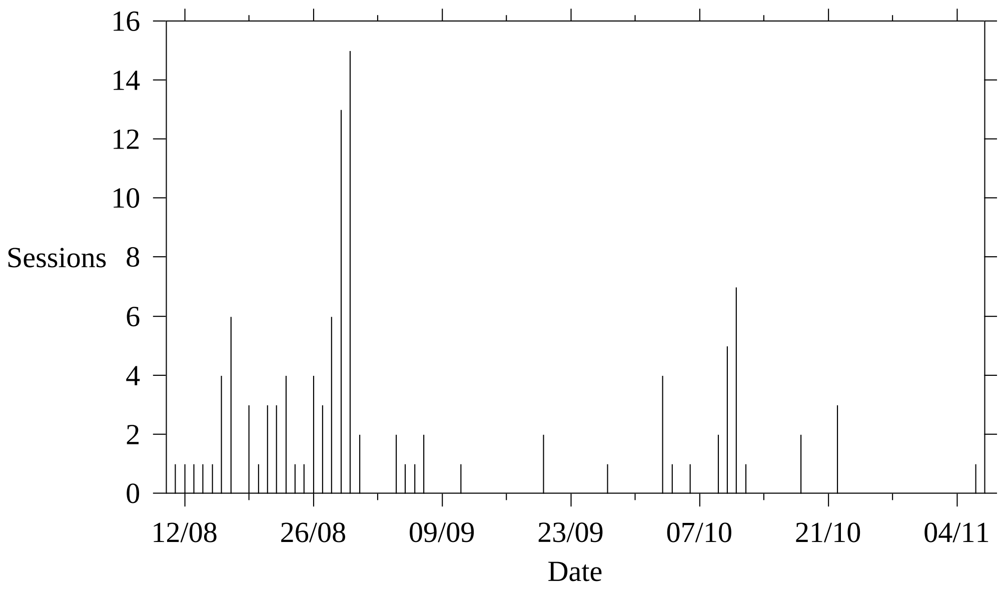


Figure B.2: Sessions of Group 9 per day, 11/Aug/2000–6/Nov/2000

```
Aug 20 2000            1
Aug 21 2000            3
Aug 22 2000            3
Aug 23 2000            4
Aug 24 2000            1
Aug 25 2000            1
Aug 26 2000            4
Aug 27 2000            3
Aug 28 2000            6
Aug 29 2000           13
Aug 30 2000           15
Aug 31 2000            2
Sep  4 2000            2
Sep  5 2000            1
Sep  6 2000            1
Sep  7 2000            2
Sep 11 2000            1
Sep 20 2000            2
Sep 27 2000            1
Oct  3 2000            4
Oct  4 2000            1
Oct  6 2000           1
Oct  9 2000           2
Oct 10 2000           5
Oct 11 2000           7
Oct 12 2000           1
Oct 18 2000           2
Oct 22 2000           3
Nov  6 2000           1
```

B.2.3 System-Level Actions in All Sessions

A total of 35 different system-level actions were performed by Group 9. The following SQL query obtained a ranked list of counts of actions:

```
SELECT Action, COUNT(*) AS Num
FROM   Log
WHERE  SessionId IN (
  SELECT DISTINCT SessionId
  FROM   Log
  WHERE  Workspace = 'cbe-group-09_Master'
  AND    NOT UserName = 'desnet-manager'
)
GROUP  BY Action
ORDER  BY 2 DESC
```

As above, the nested query obtains all session identifiers for sessions performed by members of Group 9. Output is sorted so that the most frequently performed actions are listed first, and the least frequently ones last. The graph in Figure B.3 summarizes the query result. The following output is produced by the query:

```
Action                        Num
----------------------------------
get_role_objects              683
open_object                   189
getparticipants               158
setworkspace                  157
get_user_email_homepages      152
getroles1                     151
```

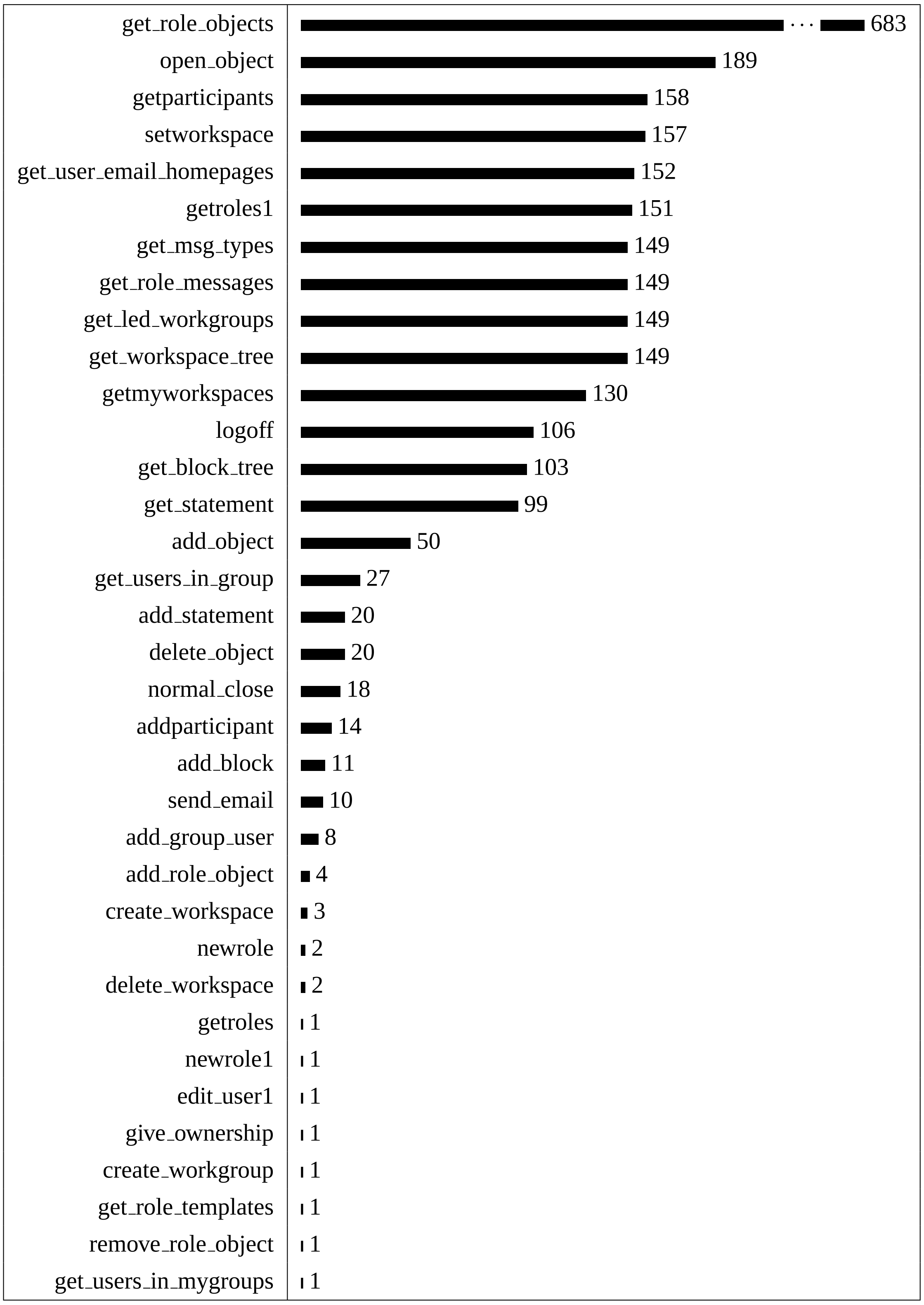


Figure B.3: Ranked list of number of occurrences of system-level actions performed by Group 9 in all sessions

```
get_msg_types                  149
get_role_messages              149
get_led_workgroups             149
get_workspace_tree             149
getmyworkspaces                130
logoff                         106
get_block_tree                 103
get_statement                   99
add_object                      50
get_users_in_group              27
add_statement                   20
delete_object                   20
normal_close                    18
addparticipant                  14
add_block                       11
send_email                      10
add_group_user                   8
add_role_object                  4
create_workspace                 3
newrole                          2
delete_workspace                 2
getroles                         1
newrole1                         1
edit_user1                       1
give_ownership                   1
create_workgroup                 1
get_role_templates               1
remove_role_object               1
get_users_in_mygroups            1
```

B.2.4 Session 4228

The following SQL query obtained all log entries for session 4228:

```
SELECT *
FROM   Log
WHERE  SessionId = 4228
ORDER  BY LogTimestamp
```

Table B.2 shows the occurrence of system-level actions in session 4228 over time. The following output is produced by the query:

```
LogId  LogTimestamp  SessionId  Workgroup  Workspace  Owner  UserName
RoleName  Action  Attrib1  Attrib2  Attrib3  Attrib4  Attrib5
Attrib6  Attrib7  Attrib8  Attrib9  Attrib10
--------------------------------------------------------------------
--------------------------------------------------------------------
------------------------------------------------------------
82905  "2000-08-26 14:06:53.773"  4228  null  null  null  cbe-shane
null  getmyworkspaces  null  null  null  null  null  null  null  null
null  null

82906  "2000-08-26 14:06:57.586"  4228  null  null  null  cbe-shane
null  setworkspace  cbe-group-09_Master  desnet-manager  null  null
null  null  null  null  null  null

82907  "2000-08-26 14:06:57.98"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_led_workgroups  null  null  null  null  null  null  null  null
null  null

82908  "2000-08-26 14:06:58.39"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_workspace_tree  cbe-group-09_Master  desnet-manager  null  null
null  null  null  null  null  null

82909  "2000-08-26 14:06:58.79"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

82910  "2000-08-26 14:06:59.19"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  BACKGROUND  null  null  null  null  null  null
null  null  null
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
add_object														X		X								X			X		X		X							
add_statement																																			X			
delete_object																						X																
get_block_tree																																	X	X		X		
get_led_workgroups			X																																			
get_msg_types												X																										
getmyworkspaces	X																																					
getparticipants										X																												
get_role_messages											X																											
get_role_objects					X	X	X	X							X		X						X		X			X		X		X						
getroles1									X																													
get_user_email_homepages													X																									
get_workspace_tree				X																																		
logoff																																						X
normal_close																																					X	
open_object																		X	X	X	X					X												
setworkspace		X																																				

Table B.2: Occurrence of system-level actions in session 4228 over time

```
82911  "2000-08-26 14:06:59.59"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  ACTION  null  null  null  null  null  null  null
null  null

82912  "2000-08-26 14:06:59.99"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DISCUSSION  null  null  null  null  null  null
null  null  null

82913  "2000-08-26 14:07:00.393"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  getroles1
null  null  null  null  null  null  null  null  null  null

82914  "2000-08-26 14:07:00.793"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
getparticipants  null  null  null  null  null  null  null  null  null
null

82915  "2000-08-26 14:07:01.193"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_messages  null  null  null  null  null  null  null  null
null  null

82916  "2000-08-26 14:07:01.593"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_msg_types  null  null  null  null  null  null  null  null  null
null

82917  "2000-08-26 14:07:01.993"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_user_email_homepages  null  null  null  null  null  null  null
null  null  null

82918  "2000-08-26 14:07:46.236"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_object
DOCUMENT  "1.0 Plan Preparation.doc"  "http://livenet.maths.uts.edu.
```

```
au/wsdocs/cbe-group-09_Master_desnet-manager/d/1.0 Plan Preparation.
doc"  null  null  null  null  null  null  null

82919  "2000-08-26 14:07:46.64"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

82949  "2000-08-26 14:14:33.943"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_object
DOCUMENT  "1.0 Plan Preparation #2.doc"  "http://livenet.maths.uts.
edu.au/wsdocs/cbe-group-09_Master_desnet-manager/d/1.0 Plan
Preparation #2.doc"  null  null  null  null  null  null  null

82950  "2000-08-26 14:14:34.346"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

82952  "2000-08-26 14:14:43.576"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  open_object
"1.0 Plan Preparation #2.doc"  null  null  null  null  null  null
null  null  null

82954  "2000-08-26 14:14:58.13"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  open_object
"1.0 Plan Preparation #2.doc"  null  null  null  null  null  null
null  null  null

82955  "2000-08-26 14:15:07.753"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  open_object
"1.0 Plan Preparation.doc"  null  null  null  null  null  null  null
null  null

82956  "2000-08-26 14:15:24.516"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  open_object
"1.0 Plan Preparation #2.doc"  null  null  null  null  null  null
```

```
null  null  null

82957  "2000-08-26 14:15:31.036"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  delete_object
"1.0 Plan Preparation #2.doc"  null  null  null  null  null  null
null  null  null

82958  "2000-08-26 14:15:31.436"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

83007  "2000-08-26 14:21:45.063"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_object
DOCUMENT  "1.0 Plan Preparation 2.doc"  "http://livenet.maths.uts.
edu.au/wsdocs/cbe-group-09_Master_desnet-manager/d/1.0 Plan
Preparation 2.doc"  null  null  null  null  null  null  null

83008  "2000-08-26 14:21:45.466"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

83009  "2000-08-26 14:21:49.17"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  open_object
"1.0 Plan Preparation 2.doc"  null  null  null  null  null  null
null  null  null

83045  "2000-08-26 14:45:37.513"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_object
DOCUMENT  "2.0 Plan Monitoring.doc"  "http://livenet.maths.uts.edu.
au/wsdocs/cbe-group-09_Master_desnet-manager/d/2.0 Plan Monitoring.
doc"  null  null  null  null  null  null  null

83046  "2000-08-26 14:45:37.913"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
```

```
null  null

83063  "2000-08-26 14:59:26.736"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_object
DOCUMENT  "3.0 Problem Identification.doc"  "http://livenet.maths.
uts.edu.au/wsdocs/cbe-group-09_Master_desnet-manager/d/3.0 Problem
Identification.doc"  null  null  null  null  null  null  null

83064  "2000-08-26 14:59:27.136"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

83067  "2000-08-26 15:12:32.166"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_object
DOCUMENT  "4.0 Propose Change.doc"  "http://livenet.maths.uts.edu.au/
wsdocs/cbe-group-09_Master_desnet-manager/d/4.0 Propose Change.doc"
null  null  null  null  null  null  null

83068  "2000-08-26 15:12:32.556"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_role_objects  DOCUMENT  null  null  null  null  null  null  null
null  null

83069  "2000-08-26 15:12:49.71"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_block_tree  cbe-group-09_Master_desnet-manager  "Discuss
Milestone"  null  null  null  null  null  null  null  null

83070  "2000-08-26 15:13:23.83"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_block_tree  cbe-group-09_Master_desnet-manager  "Discuss
Milestone"  null  null  null  null  null  null  null  null

83071  "2000-08-26 15:13:24.43"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  add_statement
1104  0  "Documents Uploaded"  "Hey there people check out the new
```

```
documents, well its a start anyway. Ciao for now."  null  null  null
null  null  null

83072  "2000-08-26 15:13:24.83"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
get_block_tree  cbe-group-09_Master_desnet-manager  "Discuss
Milestone"  null  null  null  null  null  null  null  null

83073  "2000-08-26 15:13:37.77"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member
normal_close  null  null  null  null  null  null  null  null  null
null

83074  "2000-08-26 15:13:37.98"  4228  cbe-group-09
cbe-group-09_Master  desnet-manager  cbe-shane  Member  logoff  null
null  null  null  null  null  null  null  null  null
```

Glossary & Abbreviations

ACL *abbrev. for* Access Control List.

action A function or operation that can be performed in a collaboration system. It consists of one or more *attributes* that describe it.

action context A set of information identifying the subject, referent, location, and time of an action (see *subject, referent, location, time*).

action pattern A pattern describing an action together with a particular action context. Synonymous with *pattern of virtual collaboration* (see *pattern*).

artefact A passive object or collection of objects in a collaboration system, containing information.

awareness An understanding of the activities of others, which provides a context for one's own activity (Dourish and Bellotti, 1992).

base level The level of the Information Pyramid at which the collaboration system collects data about virtual collaboration (see *Information Pyramid*).

class mapping A mapping that defines how an instance of a class representing a given concept can be created from instances of other classes.

collaboration The act of working together on a common task or process.

collaboration level The fourth level of the Information Pyramid; this is the level on which multiple users work in collaboration with each other (see *Information Pyramid*).

collaboration memory One part of an organizational memory, consisting of records of procedural aspects of collaborative activity.

collaboration process A process performed by two or more individuals working together.

collaboration space A virtual space which provides the opportunity for bringing together people, artefacts, and communication channels for individual or joint activity.

collaboration space density A measure of how many objects are contained in a single collaboration space (see *measure of a collaboration space*).

collaboration system A software system which supports virtual collaboration through the provision of collaboration spaces.

collaboration view View of information as seen from the point of view of multiple users in collaboration with each other, consisting of the objects of the collaboration system which these users interact with using the actions provided by the collaboration system (see *collaboration level*).

communication channel A facility for the exchange of messages, available to users of a collaboration space.

communication intensity A measure of the number of statements exchanged through discussion forums in a collaboration space per unit of time, calculated as an average over the history of the collaboration space (see *measure of a collaboration space*).

communication recency A measure of the number of statements "recently" exchanged through discussion forums in a collaboration space, for a pre-defined time interval (see *measure of a collaboration space*).

cooperation The joint operation or action toward a common goal or benefit.

CSCW *abbrev. for* Computer-Supported Cooperative (or Collaborative) Work.

data Uninterpreted raw facts.

document exchange intensity A measure of how often documents are exchanged in a collaboration space through create/read document actions per unit of time, calculated as an average over the history of the collaboration space (see *measure of a collaboration space*).

document exchange recency A measure of the number of "recent" document exchanges in a collaboration space, for a pre-defined time interval (see *measure of a collaboration space*).

dynamic information Information about the actions taking place in a collaboration space, representing the behaviour associated with the collaboration space's structure (see *action*).

EMOO diagram short for *extended MOO diagram* (see *extended MOO diagram*).

evolution intensity A measure of how strongly the structure of a collaboration space is subject to change, in terms of change actions per unit of time, calculated as an average over the history of the collaboration space (see *measure of a collaboration space*).

evolution recency A measure of how strongly the structure of a collaboration space has "recently" been subject to change, for a pre-defined time interval (see *measure of a collaboration space*).

extended MOO diagram An extended form of the MOO diagramming notation (see *MOO diagram*).

goal The desired realization of a specific state of a portion of the world that an activity is concerned with.

HCI *abbrev. for* Human-Computer Interaction.

HTML *abbrev. for* HyperText Markup Language.

information Interpreted data, such that it is given meaning.

Information Pyramid short for *Information Pyramid of Virtual Collaboration* (see *Information Pyramid of Virtual Collaboration*).

Information Pyramid of Virtual Collaboration A six-layer model of information related to virtual collaboration.

infrastructure level The lowest (first) level of the Information Pyramid; this is the level of the underlying software infrastructure running "below" the collaboration system (see *Information Pyramid*).

infrastructure view View of information as seen from the point of view of the infrastructure underlying the collaboration system, consisting of files that contain records of objects and actions (see *infrastructure level*).

inter-workspace map A visualization of the network of a collection of workspaces (i.e. collaboration spaces in the LIVENET system), and the relationships between them.

intra-workspace map A visualization of the internal structure of a workspace (i.e. a collaboration space in the LIVENET system).

KM *abbrev. for* Knowledge Management.

knowledge Information made actionable as a result of cognitive effort.

location The place where an action occurs (see *action context*).

macro level Levels of the Information Pyramid above the meso level; i.e. the collaboration, task and process levels (see *Information Pyramid*, *meso level*, *collaboration level, task level*, *process level*).

mapping function A function that creates instances of a specific target class (see *target class*).

measure of a collaboration space A quantitative attribute which expresses something about a certain characteristic of a collaboration space. Such a measure is derived, or computed, from information related to the collaboration space.

meso level The user level of the Information Pyramid (see *Information Pyramid*, *user level*).

micro level Levels of the Information Pyramid below the meso level; i.e. the infrastructure and system levels (see *Information Pyramid*, *meso level*, *infrastructure level*, *system level*).

MOO diagram A diagramming notation for representing internals of a task belonging to a collaboration process, showing roles, artefacts, discussions, and their relationships.

multi-artefact An artefact consisting of multiple objects (see *artefact*).

multi-role A role occupied by several people (see *role*).

object A static entity provided and maintained by a collaboration system. It consists of one or more *attributes* that describe it. The set of values of an object's attributes at a given point in time constitutes the object's *state* at that time.

OM *abbrev. for* Organizational Memory.

pattern An abstract description of the structure of a body of data.

pattern of virtual collaboration Synonymous with *action pattern* (see *action pattern*).

process A collection of related tasks with a goal, such that the accomplishment of all task goals brings about the process goal.

process level The highest (sixth) level of the Information Pyramid; this is the level on which multiple users perform collections of related tasks corresponding to processes (see *Information Pyramid*).

process model An abstract description of an actual or proposed process (Curtis et al., 1992).

process view View of information as seen from the point of view of multiple users performing processes, consisting of multiple related tasks belonging to these processes (see *process level*).

referent That which is being acted upon by an action (see *action context*).

role An organizational role occupied by one or more people.

rich picture A diagramming notation for representing a collaboration process.

session A sequence of actions performed by the same user over a given period of time, with a defined starting and ending point.

singleton artefact An artefact consisting of only one object (see *artefact*).

singleton role A role occupied by only one person (see *role*).

slot mapping A mapping that defines a correspondence between two slots.

source class In a mapping of classes, the class which is being mapped from (see *class mapping*).

source slot In a mapping of slots, the slot which is being mapped from (see *slot mapping*).

SQL *abbrev. for* Structured Query Language, a language used for retrieving and manipulating data in a relational database.

static information Information about the objects in a collaboration space, representing the collaboration space's structure (see *object*).

subject An action performer, such as a user or role or even a computational entity (see *action context*).

system level The second level of the Information Pyramid; this is the level of the collaboration system itself (see *Information Pyramid*).

system view View of information as seen from the point of view of the collaboration system, consisting of its information repositories, such as database tables and log files that contain records of objects and actions (see *system level*).

target class In a mapping of classes, the class which is being mapped to (see *class mapping*).

target slot In a mapping of slots, the slot which is being mapped to (see *slot mapping*).

task A collection of activities with a common goal, performed by one or more individuals, such that the successful completion of all the activities brings about the task's goal.

task level The fifth level of the Information Pyramid; this is the level on which multiple users perform activity corresponding to tasks (see *Information Pyramid*).

task view View of information as seen from the point of view of multiple users performing tasks, consisting of multiple collaboration-level actions and objects belonging to these tasks (see *task level*).

time The time when an action occurs (see *action context*).

URL *abbrev. for* Uniform Resource Locator, the address of a resource residing on a computer network.

user level The third level of the Information Pyramid; this is the level on which individual users operate (see *Information Pyramid*).

user view View of information as seen from the point of view of the individual user, consisting of the objects of the collaboration system which the user interacts with using the actions provided by the collaboration system (see *user level*).

virtual collaboration Collaboration which is conducted without face-to-face interaction, enabled by technology (see *collaboration*).

virtual collaboration process A collaboration process performed without face-to-face interaction, enabled by technology (see *virtual collaboration*).

virtual team A group of people who interact through interdependent tasks guided by a common purpose, working across space, time, and organizational boundaries (Lipnack and Stamps, 1997).

workspace Synonymous term for *collaboration space* as used in the LIVENET system (see *collaboration space*).

web The World-Wide Web.

XML *abbrev. for* Extensible Markup Language.

Associated Publications

Biuk-Aghai, R. P. (2000a). Virtual workspaces for web-based emergent processes. In *Fourth Pacific Asia Conference on Information Systems: Electronic Commerce and Web-Based Information Systems*, pages 864–880, Hong Kong, China.

Biuk-Aghai, R. P. (2000b). Visualization of web-based workspace structures. In Li, Q., Ozsoyoglu, Z. M., Wagner, R., Kambayashi, Y., and Zhang, Y., editors, *Proceedings of the 1st International Conference on Web Information Systems Engineering*, volume 1, pages 302–309, Hong Kong, China. IEEE Computer Society Press.

Biuk-Aghai, R. P. (2001). Visualizing structural and behavioural aspects of virtual collaboration. In *Proceedings of the Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 279–284, Cambridge, Massachusetts, USA. IEEE Computer Society Press.

Biuk-Aghai, R. P. (2003). An information model of virtual collaboration. In *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration (IRI-2003)*, pages 129–136, Las Vegas, Nevada, USA. IEEE Systems, Man, and Cybernetics Society.

Biuk-Aghai, R. P. and Hawryszkiewycz, I. T. (1999). Analysis of virtual workspaces. In Kambayashi, Y. and Takakura, H., editors, *Database Applications in Non-Traditional Environments '99*, pages 325–332, Kyoto, Japan. IEEE Computer Society Press.

Biuk-Aghai, R. P. and Simoff, S. J. (2001). An integrative framework for knowledge extraction in collaborative virtual environments. In Ellis, S., Rodden, T., and Zigurs, I., editors, *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 61–70, Boulder, Colorado, USA. ACM Press.

Biuk-Aghai, R. P. and Simoff, S. J. (2002). Assisting the design of virtual work processes via on-line reverse engineering. In *Proceedings of the 35th Hawaii Interna-*

tional Conference on System Sciences, pages 58–67, Big Island, Hawaii, USA. IEEE Computer Society Press.

Biuk-Aghai, R. P., Simoff, S. J., and Slembek, I. (2002). Knowledge-assisted reverse engineering of virtual work processes. *Informatik/Informatique*, (1/2002):30–34. Special Issue on Knowledge Management and Information Technology.

Simoff, S. J. and Biuk-Aghai, R. P. (2001a). Data mining in collaborative virtual environments: An integrating framework. In *Proceedings of WebNet 2001—World Conference on the WWW and Internet*, pages 1120–1125, Orlando, Florida, USA.

Simoff, S. J. and Biuk-Aghai, R. P. (2001b). Multimedia mining of collaborative virtual workspaces: An integrative framework for extracting and integrating collaborative process knowledge. In Zaïane, O. R. and Simoff, S. J., editors, *Proceedings of the Second International Workshop on Multimedia Data Mining (MDM/KDD'2001)*, pages 21–30, San Francisco, California, USA.

Simoff, S. J. and Biuk-Aghai, R. P. (2002a). Data mining of collaborative virtual workspaces: The ”Space-Data-Memory” framework. *Informatik/Informatique*, (1/2002):35–38. Special Issue on Knowledge Management and Information Technology.

Simoff, S. J. and Biuk-Aghai, R. P. (2002b). Discovering emergent virtual work processes in collaborative systems. In Meersman, R., Tari, Z., et al., editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE: Confederated International Conferences CoopIS, DOA, and ODBASE 2002 Proceedings*, volume 2519 of *Lecture Notes in Computer Science*, pages 286–303. Springer-Verlag.

Bibliography

Ackerman, M. S. (1994). Augmenting the organizational memory: A field study of Answer Garden. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 243–252, Chapel Hill, North Carolina, USA. ACM Press.

Ackerman, M. S. and Halverson, C. A. (2000). Reexamining organizational memory. *Communications of the ACM*, 43(1):58–64.

Ackerman, M. S. and McDonald, D. W. (1996). Answer Garden 2: Merging organizational memory with collaborative help. In *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work*, pages 97–105. ACM Press.

Ackoff, R. L. (1971). Towards a system of system concepts. *Management Science*, 17(11):661–671.

Ackoff, R. L. (1996). On learning and the systems that facilitate it. *Center for Quality of Management Journal*, 5(2):27–35.

Adams, J., Koushik, S., Vasudeva, G., and Galambos, G. (2001). *Patterns for e-business: A Strategy for Reuse*. IBM Press.

Alavi, M. and Leidner, D. E. (1999). Knowledge management systems: Issues, challenges, and benefits. *Communications of the Association for Information Systems*, 1(7).

Anand, V., Manz, C. C., and Glick, W. H. (1998). An organizational memory approach to information management. *Academy of Management Review*, 23(4):796–809.

Ansari, S., Kohavi, R., Mason, L., and Zheng, Z. (2000). Integrating e-commerce and data mining: Architecture and challenges. In *WEBKDD 2000 Workshop: Web Mining for E-Commerce – Challenges and Opportunities*, Boston, Massachusetts, USA.

Appelt, W. (1999). WWW based collaboration with the BSCW system. In *Proceedings of SOFSEM'99*, volume 1725 of *Lecture Notes in Computer Science*, pages 66–78, Milovy, Czech Republic. Springer-Verlag.

Avison, D. E. and Fitzgerald, G. (1988). *Information Systems Development: Methodologies, Techniques and Tools*. Information Systems Series. Blackwell Scientific Publications, Oxford, UK.

Baker, E., Hawryszkiewycz, I., and Rura-Polley, T. (1999). Electronic workspace networks: Tools for facilitating the creation and sharing of knowledge. In *Proceedings of KNOW'99: Deciphering Knowledge Management*, volume 1, pages 54–69, Sydney, Australia.

Bannon, L. J. and Kuutti, K. (1996). Shifting perspectives on organizational memory: From storage to active remembering. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, volume 3, pages 156–167. IEEE Computer Society Press.

Benford, S., Snowdon, D., Colebourne, A., O'Brien, J., and Rodden, T. (1997). Informing the design of collaborative virtual environments. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pages 71–80, Phoenix, Arizona, USA. ACM Press.

Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J., and Woetzel, G. (1997). Basic support for cooperative work on the World Wide Web. *International Journal of Human-Computer Studies*, 46(6):827–846.

Bentley, R., Horstmann, T., Sikkel, K., and Trevor, J. (1995). Supporting collaborative information sharing with the World Wide Web: The BSCW shared workspace system. *The World Wide Web Journal*, (1):63–74. Proceedings of the 4th International WWW Conference.

Bhaskar, R., Lee, H. S., Levas, A., Pétrakian, R., Tsai, F., and Tulskie, B. (1994). Analyzing and re-engineering business processes using simulation. In *Proceedings of the 1994 Conference on Winter Simulation*, pages 1206–1213.

Billinghurst, M. and Kato, H. (2002). Collaborative augmented reality. *Communications of the ACM*, 45(7):64–70.

Biuk-Aghai, R. P. and Simoff, S. J. (2001). An integrative framework for knowledge extraction in collaborative virtual environments. In Ellis, S., Rodden, T., and Zigurs, I., editors, *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 61–70, Boulder, Colorado, USA. ACM Press.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley.

Borning, A. and Travers, M. (1991). Two approaches to casual interaction over computer and video networks. In *Proceedings of CHI 91: Conference on Human Factors in Computing*, pages 13–19, New Orleans, Louisiana, USA. ACM Press.

Briggs, R. O., de Vreede, G.-J., Nunamaker Jr., J. F., and Tobey, D. (2001). ThinkLets: Achieving predictable, repeatable patterns of group interaction with group support systems (GSS). In *Proceedings of the Thirty-Fourth Hawaii International Conference on System Sciences*, pages 436–444, Maui, Hawaii, USA. IEEE Computer Society Press.

Çapin, T. K., Pandzic, I. S., Magnenat-Thalmann, N., and Thalmann, D. (1999). *Avatars in Networked Virtual Environments*. John Wiley & Sons, Chichester, UK.

Carr, N. G., editor (2001). *The Digital Enterprise: How to Reshape Your Business for a Connected World*. Harvard Business School Press, Boston, Massachusetts, USA.

Casati, F. and Pozzi, G. (1999). Modeling exceptional behaviors in commercial workflow management systems. In *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems*, pages 127–138, Edinburgh, Scotland. IEEE Computer Society Press.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., and Rice, J. P. (1998). Open Knowledge Base Connectivity 2.0.3. Technical Report KSL-98-06, Knowledge Systems Laboratory, Stanford University.

Checkland, P. B. (1981). *Systems Thinking, Systems Practice*. John Wiley & Sons, Chichester, UK.

Churchill, E. F. and Bly, S. (1999). Virtual environments at work: Ongoing use of MUDs in the workplace. In *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, pages 99–108, San Francisco, California, USA.

Clauer, C., Atkins, D., Weymouth, T., Olson, G., Niciejewski, R., Finholt, T., Prakash, A., Rasmussen, C., Killeen, T., Rosenberg, T., Detrick, D., Kelly, J., Zambre, Y., Heinselman, C., Stauning, P., Friis-Christtensen, E., , and Mende, S. (1995). A prototype atmospheric research collaboratory (UARC). In Szuszczewicz, E., editor, *Applications of Data Handling and Visualization Technique in Space Atmospheric Sciences*, NASA SP-519, pages 105–112.

Coleman, D. and Khanna, R., editors (1995). *Groupware: Technology and Applications*. Prentice Hall.

Conklin, E. J. (1993). Capturing organizational memory. In Baecker, R. M., editor, *Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration*, pages 561–565. Morgan Kaufmann Publishers.

Coplien, J. O. (1995). A development process generative pattern language. In Coplien, J. O. and Schmidt, D. C., editors, *Pattern Languages of Program Design*, chapter 13, pages 183–237. Addison-Wesley, Reading, Massachusetts, USA.

Corbett, J. M., Faia-Correia, M., Patriotta, G., and Brigham, M. (1999). Back up the organisation: How employees and information systems re-member organisational practice. In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*. IEEE Computer Society Press.

Curtis, B., Kellner, M. I., and Over, J. (1992). Process modeling. *Communications of the ACM*, 35(9):75–90.

D'Aveni, R. A. (1994). *Hypercompetition: Managing the Dynamics of Strategic Management*. Free Press, New York.

de Moor, A. (1999). *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*. PhD thesis, Tilburg University, The Netherlands.

Debenham, J. K. (1999). A multi-agent system for emergent process management. In *Proceedings Nineteenth International Conference on Knowledge Based Systems and Applied Artificial Intelligence, ES'99: Applications and Innovations in Expert Systems VII*, pages 51–62, Cambridge, UK.

Dietz, J. L. (1994). Modelling business processes for the purpose of redesign. In Glasson, B., Hawryszkiewycz, I., Underwood, B., and Weber, R., editors, *Business Process Re-Engineering: Information Systems Opportunities and Challenges*, pages 233–242. North-Holland.

Donlon, J. (1997). The virtual organization. *Chief Executive*, 125:58–66.

Dourish, P. (1997). Extending awareness beyond synchronous collaboration. In *CHI'97 Workshop on Awareness in Collaborative Systems*, Atlanta, Georgia, USA.

Dourish, P. and Bellotti, V. (1992). Awareness and coordination in shared workspaces. In *Proceedings ACM Conference on Computer-Supported Cooperative Work*, pages 107–114, Toronto, Canada.

Dourish, P. and Bly, S. (1992). Portholes: Supporting awareness in a distributed work group. In *Proceedings of ACM Conference on Human Factors in Computer Systems CHI'92*, pages 541–547, Monterey, California, USA.

Easteal, C. and Davies, G. (1989). *Software Engineering: Analysis and Design*. The McGraw-Hill International Series in Software Engineering. McGraw-Hill, Maidenhead, Berkshire, England.

Ellis, C. A., Gibbs, S. J., and Rein, G. L. (1991). Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):39–58.

Emmerich, W. and Gruhn, V. (1991). FUNSOFT nets: A Petri-net based software process modeling language. In *Proceedings of the 6th International Workshop on Software Specification and Design*, pages 175–184, Como, Italy. IEEE Computer Society Press.

Erickson, T. (2000). Supporting interdisciplinary design: Towards pattern languages for workplaces. In Luff, P., Hindmarsh, J., and Heath, C., editors, *Workplace Studies: Recovering Work Practice and Informing Systems Design*, pages 252–261. Cambridge University Press.

Eriksson, H. (2001). JessTab: Integrating Protégé and Jess. Online at: `http://www.ida.liu.se/~her/JessTab/`.

Farshchian, B. A. and Divitini, M. (1997). ICE: A highly tailorable system for building collaboration spaces on the WWW. In *ACM GROUP'97 Workshop on Tailorable Groupware: Issues, Methods, and Architectures*, Phoenix, Arizona, USA.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Fitzpatrick, G., Kaplan, S., and Mansfield, T. (1996). Physical spaces, virtual places and social worlds: A study of work in the virtual. In Ackerman, M. S., editor, *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work*, pages 334–343, Boston, Massachusetts, USA. ACM Press.

Fitzpatrick, G. A. (1998). *The Locales Framework: Understanding and Designing for Cooperative Work*. PhD thesis, Department of Computer Science and Electrical Engineering, The University of Queensland.

Friedman-Hill, E. J. (2001). Jess, the expert system shell for the Java platform. Technical Report SAND98-8206, Sandia National Laboratories, Livermore, California, USA. Version 6.1a3, online at: `http://herzberg.ca.sandia.gov/jess/docs/61/`.

Furst, S., Blackburn, R., and Rosen, B. (1999). Virtual team effectiveness: A proposed research agenda. *Information Systems Journal*, 9(4):249–269.

Fussell, S. R., Kraut, R. E., Lerch, F. J., Scherlis, W. L., McNally, M. M., and Cadiz, J. J. (1998). Coordination, overload and team performance: Effects of team communication strategies. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, pages 275–284, Seattle, Washington, USA. ACM Press.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Gershon, N. and Page, W. (2001). What storytelling can do for information visualization. *Communications of the ACM*, 44(8):31–37.

Giarratano, J. and Riley, G. (1998). *Expert Systems: Principles and Programming*. PWS Publishing Company, 3rd edition.

Goldman, S. L., Nagel, R. N., and Preiss, K. (1995). *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer*. Van Nostrand Reinhold, New York, NY, USA.

Greenberg, S. and Rounding, M. (2001). The notification collage: Posting information to public and personal displays. In *Proceedings of the SIG-CHI Conference on Human Factors in Computing Systems*, pages 514–521, Seattle, Washington, USA. ACM Press.

Greenhalgh, C. (1999). *Large Scale Collaborative Virtual Environments*. Springer-Verlag, London, UK.

Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. Report KSL-93-04, Stanford Knowledge Systems Laboratory.

Gruninger, M. and Lee, J. (2002). Ontology applications and design. *Communications of the ACM*, 45(2):39–41.

Gutwin, C. and Greenberg, S. (1995). Workspace awareness in real-time distributed groupware. Report 95-575-27, Dept. of Computer Science, University of Calgary, Calgary, Canada.

Gutwin, C. and Greenberg, S. (1998a). Design for individuals, design for groups: Tradeoffs between power and workspace awareness. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 207–216, Seattle, Washington, USA.

Gutwin, C. and Greenberg, S. (1998b). Effects of awareness support on groupware usability. In *Proceedings of the CHI'98 Conference on Human Factors in Computing Systems*, pages 511–518. ACM Press.

Gutwin, C. and Greenberg, S. (2002). A descriptive framework of workspace awareness for real-time groupware. *Journal of Computer-Supported Cooperative Work*, 11(3–4):411–446. Special Issue on Awareness in CSCW.

Gutwin, C., Stark, G., and Greenberg, S. (1995). Support for workspace awareness in educational groupware. In *Proceedings of the ACM Conference on Computer Supported Collaborative Learning*, pages 147–156, Bloomington, Indiana, USA.

Habermas, J. (1981). *Theorie des kommunikativen Handelns*. Suhrkamp.

Harrison, N. B. and Coplien, J. O. (1996). Patterns of productive software organizations. *Bell Labs Technical Journal*, 1(1):138–145.

Harrison, S. and Dourish, P. (1996). Re-place-ing space: The roles of place and space in collaborative environments. In Ackerman, M. S., editor, *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work*, pages 67–76, Boston, Massachusetts, USA. ACM Press.

Hawryszkiewycz, I. T. (1999a). The importance of processes. Unpublished note.

Hawryszkiewycz, I. T. (1999b). Workspace networks for knowledge sharing. In Debrency, R. and Ellis, A., editors, *Proceedings of AusWeb99, the Fifth Australian World Wide Web Conference*, pages 219–227, Ballina, Australia.

Hawryszkiewycz, I. T. (2000). Analysis for cooperative business processes. In Zowghi, D., editor, *Proceedings of the Fifth Australian Workshop on Requirements Engineering*, pages 3–11, Brisbane, Australia.

Herrmann, T., Hoffmann, M., Loser, K.-U., and Moysich, K. (2000). Semistructured models are surprisingly useful. In *Proceedings of Coop 2000*, pages 159–174, Sophia Antipolis, France.

Hiltz, S. R. and Turoff, M. (1985). Structuring computer-mediated communication systems to avoid information overload. *Communications of the ACM*, 28(7):680–689.

Hollingsworth, D. (1995). The workflow reference model. Document TC00-1003, Workflow Management Coalition.

Holt, A. W. (1988). Diplans: a new language for the study and implementation of coordination. *ACM Transactions on Information Systems*, 6(2):109–125.

Holt, A. W. (1997). *Organized Activity and Its Support by Computer*. Kluwer Academic Publishers.

Huang, M. L., Eades, P., and Wang, J. (1998). On-line animated visualization of huge graphs using a modified spring algorithm. *Journal of Visual Languages and Computing*, 9(6):623–645.

Huber, G. P. (1996). Organizational learning: The contributing processes and the literatures. In Cohen, M. D. and Sproull, L. S., editors, *Organizational Learning*, pages 124–162. Sage Publications.

Johansen, R., Sibbet, D., Benson, S., Martin, A., Mittman, R., and Saffo, P. (1991). *Leading Business Teams: How Teams Can Use Technology and Group Process Tools to Enhance Performance*. Addison-Wesley.

Katzy, B., Evaristo, R., and Zigurs, I. (2000). Knowledge management in virtual projects: A research agenda. In *Proceedings of the Thirty-Third Hawaii International Conference on System Sciences*, pages 121–129. IEEE Computer Society Press.

Keller, G., Nüttgens, M., and Scheer, A.-W. (1992). Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Research Report 89, Institut für Wirtschaftsinformatik, Universität des Saarlandes, Saarbrücken, Germany. (“Semantic Process Modelling Based on Event-Driven Process Chains (EPK)”, in German).

Kraut, R., Egido, C., and Galegher, J. (1988). Patterns of contact and communication in scientific research collaboration. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 1–12, Portland, Oregon, USA. ACM Press.

Kueng, P., Bichler, P., and Kawalek, P. (1996). How to compose an object-oriented business process model? IPG working paper, Informatics Process Group, University of Manchester, UK.

Landry, J. R. (1999). Forgetful or bad memory? In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*. IEEE Computer Society Press.

Lee, J. H., Prakash, A., Jaeger, T., and Wu, G. (1996). Supporting multi-user, multi-applet workspaces in CBE. In Ackerman, M. S., editor, *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work*, pages 334–343, Boston, Massachusetts, USA. ACM Press.

Lipnack, J. and Stamps, J. (1997). *Virtual Teams: Reaching Across Space, Time, and Organizations with Technology*. Wiley, New York.

Lipnack, J. and Stamps, J. (1999). Virtual teams. *Executive Excellence*, 16(5):14–15.

Maier, R. and Lehner, F. (2000). Perspectives on knowledge management systems – theoretical framework and design of an empirical study. In Hansen, H., Bichler, M., and Mahrer, H., editors, *Proceedings of the 8th European Conference on Information Systems ECIS 2000*, pages 685–693, Vienna, Austria.

Mansfield, T., Kaplan, S., Fitzpatrick, G., Phelps, T., Fitzpatrick, M., and Taylor, R. (1997). Evolving Orbit: A progress report on building locales. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pages 241–250, Phoenix, Arizona, USA. ACM Press.

Mansfield, T., Kaplan, S., Fitzpatrick, G., Phelps, T., Fitzpatrick, M., and Taylor, R. (1999). Toward locales – supporting collaboration with Orbit. *Information and Software Technology*, 41(6):367–382.

Martin, D., Rodden, T., Rouncefield, M., Sommerville, I., and Viller, S. (2001). Finding patterns in the fieldwork. In Prinz, W., Jarke, M., Rogers, Y., Schmidt, K., and Wulf, V., editors, *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*, pages 39–58, Bonn, Germany. Kluwer Academic Publishers.

Moorman, C. and Miner, A. S. (1998). Organizational improvisation and organizational memory. *Academy of Management Review*, 23(4):698–723.

Moran, T. P. and Anderson, R. (1990). The workaday world as a paradigm for CSCW design. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 381–393, Los Angeles, California, USA. ACM Press.

Morrison, J. and Olfman, L. (1999). Organizational memory and knowledge management (minitrack introduction). In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*, page 564. IEEE Computer Society Press.

Naff, K. C. (1995). Hypercompetition drives business into the 21st century. *Business Credit*, 97(4):28–29.

Niederman, F. and Beise, C. M. (1999). Defining the ”virtualness” of groups, teams, and meetings. In Prasad, J., editor, *Proceedings of the 1999 ACM SIGCPR Conference on Computer Personnel Research*, pages 14–18, New Orleans, Louisiana, USA. ACM Press.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1):14–37.

Noy, N. F., Fergerson, R. W., and Musen, M. A. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *2nd International Conference on Knowledge Engineering and Knowledge Management (EKAW‘2000)*, volume 1937 of *Lecture Notes in Computer Science*, pages 17–32, Juan-les-Pins, France. Springer-Verlag.

Nunamaker Jr., J. F., Romano Jr., N. C., and Briggs, R. O. (2001). A framework for collaboration and knowledge management. In *Proceedings of the Thirty-Fourth Hawaii International Conference on System Sciences*, pages 461–472, Maui, Hawaii, USA. IEEE Computer Society Press.

Pankoke-Babatz, U. and Syri, A. (1997). Collaborative workspaces for time deferred electronic cooperation. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pages 187–196, Phoenix, Arizona, USA. ACM Press.

Picot, A. and Reichwald, R. (1987). *Bürokommunikation—Leitsätze für den Anwender*. AIT, Hallbergmoos, Germany, 3rd edition. (“Office Communication—Guidelines for the User”, in German).

Poltrock, S. E. and Engelbeck, G. (1997). Requirements for a virtual collocation environment. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pages 61–70, Phoenix, Arizona, USA. ACM Press.

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58.

Rhee, I. (1999). Support for global teams. *IEEE Internet Computing*, 3(2):30–31.

Richards, H. D. and Makatsoris, H. G. (2002). The metamorphosis to dynamic trading networks and virtual corporations. In Franke, U. J., editor, *Managing Virtual Web Organizations in the 21st Century: Issues and Challenges*, chapter IV, pages 61–89. Idea Group Publishing.

Robertson, G. G., Card, S. K., and Mackinlay, J. D. (1993). Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4):57–71.

Rohloff, M. (1996). Reference model and object oriented approach for business process design and workflow management. In *Proceedings of the Information Systems Conference of New Zealand*, pages 43–52.

Roseman, M. and Greenberg, S. (1996). TeamRooms: Network places for collaboration. In *Proceedings of the ACM CSCW'96 Conference on Computer-Supported Cooperative Work*, pages 325–333. ACM Press.

Sa, J., Warboys, B., and Keane, J. (1993). OBM: A specification method for modelling organisational process. In *Proceedings of Workshop on Constraint Processing CSAM '93*, pages 143–158, St. Petersburg, Russia.

Sadagic, A., Towles, H., Holden, L., Daniilidis, K., and Zeleznik, B. (2001). Tele-immersion portal: Towards an ultimate synthesis of computer graphics and computer vision systems. In *4th Annual International Workshop on Presence*, Philadelphia, Pennsylvania, USA.

Sadiq, S. W. (1999). Workflows in dynamic environments—can they be managed? In Zhang, Y., Rusinkiewicz, M., and Kambayashi, Y., editors, *Cooperative Databases and Applications '99—The Proceedings of the 2nd International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'99)*, pages 178–189, Wollongong, Australia.

Sarin, S. K., Abbott, K. R., and McCarthy, D. R. (1991). A process model and system for supporting collaborative work. In *Proceedings of the Conference on Supporting Group Work*, pages 213–224, Atlanta, Georgia USA. ACM Press.

Sawhney, M. and Parikh, D. (2001). Where value lives in a networked world. *Harvard Business Review*, 79(1):79–86.

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley Longman, 3rd edition.

Simoff, S. J. and Maher, M. L. (2000). Analysing participation in collaborative design environments. *Design Studies*, 21(2):119–144.

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, California, USA.

Spellman, P. J., Mosier, J. N., Deus, L. M., and Carlson, J. A. (1997). Collaborative virtual workspace. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pages 197–203, Phoenix, Arizona, USA. ACM Press.

Spiegler, I. (2000). Knowledge management: A new idea or a recycled concept? *Communications of the Association for Information Systems*, 3(14).

Spiliopoulou, M. and Pohle, C. (2001). Data mining for measuring and improving the success of web sites. *Data Mining and Knowledge Discovery*, 5(1/2):85–114.

Steinfield, C. (2002). Realizing the benefits of virtual teams. *Computer*, 35(3):104–106.

Syring, M. and Hasenkamp, U. (1997). Communication-orientated approaches to support multi-user processes in office work. In Kirn, S. and O'Hare, G., editors, *Cooperative Knowledge Processing: The Key Technology for Intelligent Organizations*, pages 43–63. Springer-Verlag, London.

Tanenbaum, A. S. (1988). *Computer Networks*. Prentice Hall, 2 edition.

ter Hofte, G., van der Lugt, H. J., and Houtsma, M. A. (1996). Co^4: A comprehensive model of groupware functionality. In Verbraeck, A., editor, *APTEC Conference of Euromedia'96*, pages 231–238, London, UK. Society for Computer Simulation International, London.

Tuomi, I. (1999). Data is more than knowledge: Implications of the reversed knowledge hierarchy for knowledge management and organizational memory. In *Proceedings of the Thirty-Second Annual Hawaii International Conference on System Sciences*. IEEE Computer Society Press.

Vail III, E. F. (1999). Knowledge mapping: Getting started with knowledge management. *Information Systems Management*, 16(4):16–23.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51.

Walsh, J. P. and Ungson, G. R. (1991). Organizational memory. *Academy of Management Review*, 16(1):57–91.

Weigand, H., de Moor, A., and van den Heuvel, W.-J. (2000). Supporting the evolution of workflow patterns for virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 6, Hawaii, USA. IEEE Computer Society.

Wessner, M. and Pfister, H.-R. (2000). Points of Cooperation: Integrating cooperative learning into web-based courses. In *Proceedings of the NTCL2000, The International Workshop for New Technologies for Collaborative Learning*, pages 33–41, Hyogo, Japan.

Yu, E. S. and Mylopoulos, J. (1993). An actor dependency model of organizational work – with application to business process reengineering. In *Proceedings of the Conference on Organizational Computing Systems*, pages 258–268. ACM Press.

Zukunft, O. and Rump, F. (1996). From business process modelling to workflow management: An integrated approach. In Scholz-Reiter, B. and Stickel, E., editors, *Business Process Modelling*, pages 3–22. Springer-Verlag.