Survey of computer-supported collaboration in support of business processes

Carl K. Chang*

Department of Computer Science,
Iowa State University,
Ames, IA 50011, USA
E-mail: chang@cs.iastate.edu
*Corresponding author

Jia Zhang

Department of Computer Science,
Northern Illinois University,
DeKalb, IL 60115, USA
E-mail: jiazhang@cs.niu.edu

Kai H. Chang

Department of Computer Science and Software Engineering,
Auburn University,
Auburn, AL 36849, USA
E-mail: kchang@eng.auburn.edu

Abstract: Business process via collaboration can benefit from the Computer-Supported Cooperative Work (CSCW) community. The purpose of this paper is to provide a survey of CSCW research to examine the current IT-based collaboration techniques capable of supporting business processes. The CSCW field is categorised by the issues tackled by researchers. Each category is analysed based on past approaches and present achievements; trends for future research and development are then predicted. Finally, we discuss how CSCW research can facilitate distributed business process integration and management.

Keywords: business process; collaboration; Computer-Supported Cooperative Work (CSCW); control; coordination; policy.

Reference to this paper should be made as follows: Chang, C.K., Zhang, J. and Chang, K.H. (XXXX) 'Survey of computer-supported collaboration in support of business processes', *Int. J. Business Process Integration and Management*, Vol. X, No. Y, pp.XXX–XXX.

Biographical notes: Carl K. Chang (PhD) is a Professor and Chair in the Department of Computer Science at Iowa State University. His research interests include requirements engineering, software architecture and net-centric computing. He is a founding member of the IEEE International Requirements Engineering Conference (RE) and served as the General Chair of ICRE2000 and RE2003. He also chaired the steering committee for the 2004 IEEE-CS/IPSJ International Symposium on Applications and the Internet (SAINT) after serving as the Programme Chair of SAINT2002 and General Chair of SAINT2003. In 2005, he was the General Chair of IEEE International Conference on Web Services (ICWS) and IEEE International Conference on Services Computing (SCC). He is also active in the educational activities and spearheaded the Computing Curricula 2001 (CC2001) project jointly sponsored by the IEEE Computer Society, ACM and the National Science Foundation. He served as the Editor-in-Chief for IEEE Software in 1991–1994. He is a fellow of IEEE, a fellow of AAAS and the President of the IEEE Computer Society in 2004.

Jia Zhang (PhD) is an Assistant Professor in the Department of Computer Science at Northern Illinois University and also a Guest Scientist of National Institute of Standards and Technology (NIST). Her current research interests centre around software trustworthiness in the domain of web services, with a focus on reliability, integrity, security and interoperability. She has published more than 60 technical papers in journals and conference proceedings. She also has seven years of industrial experience as software technical lead in web application development. She received her PhD in Computer Science from the University of Illinois in Chicago in 2000. She is a member of the IEEE and ACM.

Kai H. Chang (PhD) is an Alumni Professor of Computer Science and Software Engineering at Auburn University. His research interests include Computer-Supported Cooperative Work (CSCW), software testing, software metrics and software quality and security.

He received a Diploma in Electrical Engineering from the Taipei Institute of Technology and his MS and PhD in Electrical and Computer Engineering from the University of Cincinnati. He is a member of the ACM and IEEE.

1 Introduction

In modern business society, the workforce has been becoming increasingly distributed. Non-collocated corporate resources with broader expertise need to work together to address increasingly complex business needs. As some members of organisations are remotely located, it has been necessary for the people to travel, incurring high travel costs. These costs include both 'hard' and 'soft' costs (HardSoftCost, 2002). Hard costs refer to direct out-of-pocket expenditures such as airline travel, food and lodging. Soft costs refer to personnel labour time. The significance of business travel-related costs is illustrated by MCI (HardSoftCost, 2002), a global leader in business communications, in their annual survey *Meetings in America.* The 2001 survey conducted by InfoCom revealed that the average travel-related cost per person per business collaboration for a Fortune 500 corporation was $527; the cost was $547 for a Fortune 500–2000 corporation and the cost was $412 for other corporations. As an example, for a five-person business meeting with four attendees travelling by air, the combined hard and soft costs were $5197.50.

The 11 September 2001 terrorist attack on the World Trade Center forced US corporations to reduce the travel allowances. MCI's 2002 report revealed that 25% of business travellers had reduced their air travel by March 2002 (MCI2002, 2002). Even if travel allowances rebound, there are other essential aspects that cause people to be reluctant to travel. According to a national study released on 20 October 2003 by MCI, the top reasons for employees not travelling were ranked as follows:

1 time concern and efficiency (69%)

2 reduced corporate travel budgets (37%)

3 company policy (36%)

4 better work-life balance (29%)

5 increased productivity (28%) and

6 concerns about travel safety due to diseases (e.g. SARS) and terrorism (12%) (MCI2003, 2003).

As a result, US organisations have been gradually adopting virtual collaboration technology or so-called Computer-Supported Cooperative Work (CSCW), to support distributed business process (Hodel et al., 2004). This technology offers people a promising option of not being physically present at a business collaboration, which allows them to be more productive while maintaining relationships, thus creating a better balance of work and personal activities. However, since its inception in 1984 (Grudin and Poltrock, 1997), the CSCW domain has been a rich area of inquiry with about 20 years of research and implementations. The breadth of this research and solutions can be bewildering to a business practitioner seeking a custom solution to her needs of better business process collaborations.

Therefore, this paper intends to perform a broad review of the CSCW domain to provide business practitioners a guide to the CSCW field. The remainder of this paper is organised as follows. In Section 2, we provide an overview of the CSCW field. In Section 3, we introduce CSCW research challenges. In Section 4, we discuss each CSCW research issue in detail, including its past approaches, present approaches, and predicted future trends. In Section 5, we discuss how and why CSCW researches can facilitate distributed business process. Finally, in Section 6, we make conclusions.

2 Overview of CSCW

In this section, we will briefly introduce the basic concept of CSCW. CSCW is an interdisciplinary research area (Horn et al., 2004) that focuses on how to incorporate computing and networking technologies to facilitate cooperation and collaboration among people (Grudin, 1991, 1994; Mills, 1999). CSCW embraces a broad field of disciplines in computer science and engineering realms such as Human–Computer Interaction (HCI), networks, multimedia, communications, database management, distributed system, object-oriented concepts, Virtual Reality (VR), software engineering and Artificial Intelligence (AI) (Edwards, 1996; Ellis et al., 1991).

To facilitate an effective and efficient human cooperation, a 3C model acts as a fundamental basis. As shown in Figure 1, the 3C model represents a three-layer supportive mechanism, namely:

1 communication

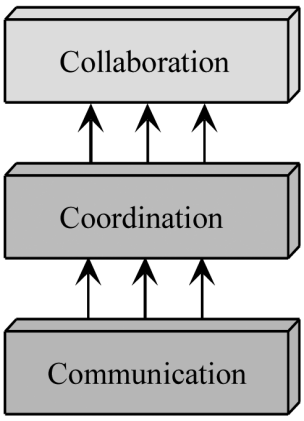
2 coordination and

3 collaboration.

Among these three layers, the communication layer provides fundamental communication channels. It is not always feasible or even desirable for all communication among collaborators to be face-to-face interactions. However, it is necessary that the communication medium and methodology replacing face-to-face interactions strengthen the sense of group, without detracting the group from collaborative tasks (Cohen, 1996). The coordination layer illustrated in the middle layer refers to the methods of helping people to work together harmoniously and managing interdependencies among processes, people and available resources (Malone and Crowston, 1990). Such a coordination requires a set of rules, either formal or informal, to regulate the conduct of the participants

(Nicollin and Sifakis, 1994). With the support of the communication and coordination layers, the top collaboration layer is the ultimate goal that requires a space to facilitate collaborators to share common and ever evolving information among them. Within such a collaboration space, the group must be aware of what others are doing with respect to the collaborative task (Ellis et al., 1991), which is often referred to as 'group awareness'.

Figure 1 Three layers in CSCW



With such a three-layer structure, human collaboration can be classified into two general categories: formal and informal. The essential difference between the two types of CSCW is whether there exists a set of formal collaboration rules to be enforced in a cooperative work. A collaboration rule is a set of well-defined perpetual steps that must be followed by each collaborator during her cooperative work until a goal is reached. A networked chatting room is a good example of informal CSCW, where people can exchange ideas arbitrarily. Through this collaboration, people may get some incentives from each other to facilitate their work; however, generally speaking, informal CSCW only provides a shared environment for people without guaranteeing the order of cooperation among people. Therefore, in our opinion, a formal CSCW is a special case of informal CSCW systems by tightening the collaboration rules and imposing stringent resource requirements.

Now that we have introduced the paradigm of CSCW, in Section 3, we will introduce the fundamental research challenges posed by CSCW.

3 CSCW research challenges

In this section, we will introduce the most significant research topics in the field of CSCW.

Since its inception in 1984 (Grudin and Poltrock 1997), the last two decades have witnessed enormous research achievements conducted in the CSCW field (Grudin, 1994). We found the most major research efforts centre around the three CSCW layers are as shown in Figure 1. In other words, in each layer, a set of fundamental research topics are identified. Figure 2 depicts an overview of the space containing CSCW research issues within the three layers.

Along the communication direction, three research topics are identified:

- concurrency control, which focuses on the management of parallel threads of HCIs that involve potential conflicts
- human proxy, which tries to utilise agent technology (Calvary et al., 1997) to facilitate human collaboration and
- environment, which refers to how to define environmental features, how to detect and adjust to environmental changes.

Along the coordination direction, we find four research topics:

- access control, which aims at controlling simultaneous accesses to a shared artefact
- social conduct, which intends to understand the nature of cooperative work
- policies, which focus on how to set up rules effectively and flexibly for control and coordination and
- coordination, which focuses on how to control and harmonise human interactions.

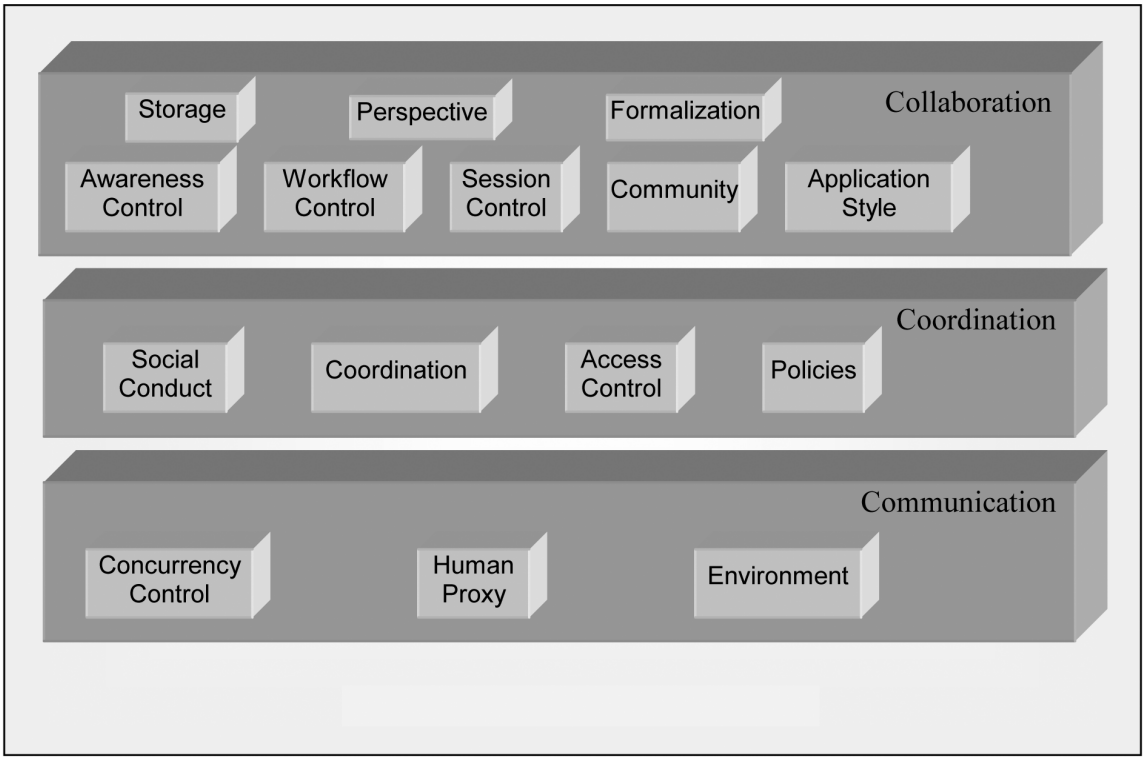
Along the collaboration dimension, we see eight major research topics:

- awareness control, which facilitates a collaborative process
- community, which focuses on how to analyse human collaboration from the perspective of social psychology
- perspective, which focuses on how to dynamically catch and manage collaboration interactions
- workflow control, which keeps correct working order of a CSCW system
- session control, which aims for regulating a piece of collaborative work
- storage, which deals with how to efficiently store, propagate and request shared information
- application style, which refers to how to reuse legacy single-user applications and whether to support synchronous and asynchronous collaborations and
- formalisation, which focuses on setting up a formal framework of a collaboration system and a tailored formal language.

From Figure 2, it is clear that these identified research topics are not completely isolated from each other. First of all, these topics are grouped in three layers, which indicates:

1 each topic represents one aspect of the corresponding layer
2 they work together to establish a layer of effective support for human collaboration and
3 each topic in a layer implies the support from other topics in the lower layers.

Figure 2 Research topics within three layers



These interrelationships indicate that when we consider an issue in the design of a CSCW-based business process, we need to consider the related issues as well. For example, when we consider the issue about access control, we shall consider the issues of concurrency, awareness and information presentation.

In addition, as shown in Figure 2, the control management in CSCW can be divided into five subjects that are located at different layers: concurrency control in the communication layer, access control in the coordination layer, session control, workflow control and awareness control in the collaboration layer. This phenomenon highlights the importance of the control management in the CSCW field, as well as the fact that it needs to be supported at different levels.

In summary, this categorisation sketches a roadmap of major CSCW research topics that are essential to the success of a CSCW system in the context of the 3C model.

4 CSCW research issues survey

A variety of research has been conducted to analyse the aspects of each research topic in CSCW research, propose possible considerations and solutions, and establish a set of customisable templates or patterns (Gamma et al., 1994) for facilitating the implementation of a real collaboration system. In this section, we will survey and analyse each topic. The strategy of our survey is as follows. For each research topic, we will examine the literature and divide the existing approaches into past and present categories; then we will predict the future trend based on the perceived paradigm shifts. It should be noted that our ultimate goal is to analyse the literature and identify the directions for the future; therefore, we do not use any distinct timeline as the only criterion to differentiate the past approaches and current ones. Instead, for every topic, we examine the corresponding publications to identify the major technical transformations and milestones. Certainly, the approaches adopted at present time are considered to be current approaches; however, they might either have emerged a few years ago or have been around for a decade. In addition, as we discussed in the previous section, as the five control managements-oriented topics are closely related, we will put them in one group for discussions.

4.1 Application style

Application style in CSCW refers to the topics of:

1 reusing legacy single-user applications and

2 whether to support synchronous and asynchronous collaborations.

In the past, applications were designed to serve single users (Li and Li, 2002); therefore, no relationship existed between two copies of an application running for different users. Meanwhile, different users run applications asynchronously.

If every application has to be constructed from scratch, it will be expensive, time-consuming and less reliable. Therefore, developers wish to reuse existing single-user applications as the basic off-the-shelf components for the new collaborative systems; with a few, if any, changes. This reusability is referred to as *collaboration transparency* in Li and Li (2002). Reusability thus becomes one of the essential issues in the CSCW realm, whose objective is to allow one single-user application to be utilised cooperatively by multiple users (Li and Li, 2002).

At the present, various applications have evolved to reuse existing single-user applications with a few modifications to serve multi-users to work synchronously or asynchronously while interacting with each other (Jeffay et al., 1992; Li and Li, 2002). Their common ground is that these applications normally provide system-level support to maintain the consistency and provide communication mechanisms between different users. These applications also bear some differences.

One difference is the way to reuse the existing single-user applications. One way is to run the single-user application at a server machine, whereas every user launches the application virtually through the server, such as software project version control systems (e.g. Concurrent Versions System (CVS), 2004; StarTeam, 2004). The other way is to incorporate the original application with some additional system-supported collaboration functions, such as real-time internet-based shared editor REDUCE (Yang et al., 2000).

Another difference is whether the shared application supports synchronous or asynchronous collaborations. These two collaboration modes differ in the way how dynamic changes of shared data are propagated among collaborators (Kouramajian, 1995). In the synchronous mode, all individual changes are reflected immediately in all participants' views. In other words, all users' observations of the shared artefact are always kept consistent. As an example, electronic distributed conferences mainly provide a synchronous team support. On the other hand, in the asynchronous mode, users may originally work independently on their own local copies and then submit their work results. Only after their work is successfully merged back to the shared working objects, the modifications will be distributed to all participants. Therefore, in this mode, each user's view of the workspace may be inconsistent from time to time. Typical examples are various software project version control systems, for example, CVS (2004) and StarTeam (2004), where each developer submits her update to a *merge* process before the changes are integrated into the project-wide shared version.

In recent years, researchers have been exploring methods to integrate synchronous and asynchronous modes into one system to provide a finer-grained collaboration. TeamSpace (Geyer et al., 2001) incorporated both synchronous and asynchronous types of collaboration into its task-oriented virtual meeting environment. With its prototype that treats meetings as events in the team's work context, members in a TeamSpace can shift smoothly among different work modes. Another system UARC utilises a room as a metaphor to support both synchronous and asynchronous collaborations (Subramanian and Malan, 1999). Consortium (Kouramajian, 1995) introduced a concept of semi-synchronous interaction mode, which proposes a more flexible mechanism that allows data to be shared in a controlled manner, that is, the users can establish different degrees of data coherency based upon, for example, Boolean expressions in terms of time and/or the state of shared objects. In a Consortium system, interactions are simply defined on the predefined object level; however, a negotiable and time-based lock for each object can support a finer-grained sharing level.

As a summary, till date CSCW systems are capable of supporting multi-users with synchronous and asynchronous collaborations. However, the current CSCW applications only support a group of users with relatively simple relationships. As more and more users are to be supported in one application to cooperate towards a common and complicated business goal, we believe that future collaboration systems will become community-based ubiquitous applications, where powerful system facilities will be provided to support comprehensive and complex hierarchical relationships and dependencies among users. Furthermore, the advancement of mobile information appliances poses a new challenge to the direction, such as how to realise ubiquitous collaboration with powerful system facilities. Research in this field will concentrate on different types of business applications associated with specific domain analysis and business rules. Some researchers have started to explore along this direction. By embedding application semantics into application sharing mechanisms, Li et al. explored the feasibility of supporting transparent sharing and interoperation of heterogeneous single-user applications (Du et al., 2003; Li and Li, 2002).

4.2 Control

Five forms of control are fundamental in a CSCW environment, including concurrency control, access control, session control, workflow control and awareness control.

4.2.1 Concurrency control

The necessity of concurrency control originally arose from managing shared resources among simultaneously running threads in operating systems (Silberschatz and Galvin, 1998). In the CSCW realm, concurrency control focuses on the management of parallel threads of HCIs that involve potential conflicts (Russel et al., 1996; Sabbir and Ravindran, 2004). Concurrency control ensures that multiple threads of related collaboration work can be incorporated in one integrated CSCW environment. In addition, in synchronous cooperative work, when different users share a common object, concurrency control is compulsory to maintain the shared object consistent (Sabbir and Ravindran, 2004). In more detail, each collaborator should see the exact same sequence of actions to be performed on the shared artefact.

For the CSCW realm, in the past, concurrency control did not catch much attention because applications were single-threaded. Currently, applications are usually running multi-threaded; therefore, concurrency control becomes critical. The concept of transaction is adopted as a result, which refers to one execution of a cooperative process (Jiang et al., 2002), that is, it is an instance of a thread of interactions. Consortium (Kouramajian, 1995) provided a framework to define customised transactions in a collaborative environment, from both structural and

behavioural perspectives. Two layers were defined in the framework:

1 a session layer provides a convenient abstraction for a collective activity and

2 a transaction layer consists of an ordered set of sessions.

In addition, CovaTM (Jiang et al., 2002) presented a comprehensive transaction model to support collaboration work, where user intervention is explicitly introduced. Sabbir and Ravindran (2004) suggested that a temporal relationship should be bound to data segments.

Currently, most of the work in this direction derives from concurrency control research in distributed systems. However, there is a significant distinction between them: a distributed system is oriented to a group of computer programs, while a CSCW system is oriented to a group of human collaborators with rich and intricate interactions. As the CSCW applications become more mature to support more complex business processes, we predict that future concurrency control needs to handle comprehensive and complicated massively parallel-threaded collaborations.

4.2.2 Access control

Access control aims at controlling simultaneous accesses to a shared artefact.

Earlier solutions focused on the specification of users' rights to access each shared object. This specification was normally simple and flat and could be expressed by a linear access matrix. COLA (Trevor et al., 1994) presented an access control adapter to guard the operations on shared objects. Whenever an object is spawned, COLA attached to the object an access control adapter, which contains the context information such as name, role and activities. The adapter will then utilise this context information, together with a set of predefined access rules, to determine whether a user's access request can be granted. Suite (Dewan and Shen, 1998) adopts an extended access matrix mechanism to control the accesses in multi-user interfaces. Its matrix supports a large set of rights, ranging from the traditional semantic rights to interaction and coupling rights. A set of inference rules is also presented to derive default permissions. However, in the traditional solutions, the access right for every user on each shared artefact is immutable, and the granularity of control is coarse.

Recent access control is extensively based on a hierarchical access matrix, which is capable of handling hierarchical structures of access privileges (Lee, 1998; Lee et al., 2001). System developers can define different fine-grained access rights on each shared artefact for each user. Preguiça et al. (2000) presented a replicated object store to provide high availability for asynchronous collaborators to independently access and modify shared artefacts. An object framework is supplied to decompose objects in several components so as to enable different object executions on different components. Begole et al. (2002) analysed the visualisations of awareness histories to study the working rhythms among distributed collaborative groups.

Some other researchers attempted to provide ubiquitous access to shared information for mobile users (Hengartner and Steenkiste, 2004; Kirda et al., 2002). The Satchel system (Lamming et al., 2000) was designed to offer easy, secure and timely access by a token-based prototype. Tokens are defined as small secure references that represent documents on mobile devices. By transmitting small tokens through the wireless channels, a large amount of transmissions of document contents is committed only on-demand. The MOTION (Kirda et al., 2002) service architecture also supports mobile teamwork. It takes into account the different connectivity modes of users, and integrates eXtensible Markup Language (XML) meta-data for distributed searches and descriptions. In addition, Hengartner and Steenkiste (2004) discussed the necessity of associating policy with access control and how to use certificate and centralised server to grant trust access. Krowne and Bazaz (2004) proposed a CSCW-oriented authority model that determines who has access control over shared artefacts and how this control manifests.

In general, however, we found that the access control technique till date still remains inflexible, since all access control requirements have to be predefined and remain immutable in the process of collaboration work. Therefore, we believe that future research needs to address how to allow privileged users to dynamically define, modify and customise access rights on-demand.

4.2.3 Session control

Session control aims for regulating a piece of collaborative work. Flexible session control allows participants to dynamically join and leave a session (Edwards, 1996; Ellis et al., 1991), to act multiple roles simultaneously, and to smoothly shift between different roles (Li and Muntz, 1998).

Session control improvement has been made to move from predefined mode to more dynamic mode. In the past, the organisation of collaboration was preregulated by organisers and could not be changed 'after the fact' (Li and Muntz, 1998). At present, organisers may still institute the session control in advance. However, tools are provided for organisers to allow people to join in or leave collaboration on-the-fly. GroupKit (Roseman and Greenberg, 1996) defined a distributed architecture to realise session control by providing replicated session managers. With the aid of GroupKit primitives, programmers are allowed to customise session control policies associated with each session manager. However, GroupKit does not provide protection to the application state (Dewan and Shen, 1998), which is critical to keep a collaborative work consistent. COCA (Li and Muntz, 1998) predefines a set of roles and explicitly defines associated coordination policies for each role. The system allows participants to dynamically join or leave the collaboration by simply taking or dropping roles. Similarly, in ADOME (Li and Lochovsky, 1998), the roles define the social commitments or obligations of members in a group. The commitments are restrictions on how members must act. When a member joins a group, she

chooses one or more roles, thereby acts according to the commitments associated with the roles.

However, how to dynamically create new roles and manage complex hierarchical sessions remains a challenge. Tools are needed for organisers to allow people to change roles or to associate with other members on-the-fly. We believe that future session control will provide more flexibility; therefore, it can be updated on-demand.

4.2.4 Workflow control

Workflow management is vital to keep the correct working order of a CSCW system, by allowing a group of people to work together towards some common goal (Haynes et al., 2004). CSCW is an environment for geographically distributed people to work together; therefore, it requires a system support to efficiently coordinate people with complex and distributed work practices. Process modelling has been developed to meet this requirement (Grundy et al., 1998), where a framework was presented to organise participants to collaborate on a common task.

Earlier workflow management used linear and centralised control. A central server keeps track of the workflow of applications, normally utilising a table-like information structure. Recent management has turned to multi-dimensional and distributed control. In a distributed collaboration, there may be many parallel tasks running at various layers in the system architecture simultaneously. This kind of control can no longer be supported by one-dimensional linear control; therefore, the responsibility of flow control is usually distributed to a group of controllers. Among these efforts, Klinker et al. (1995) use a workflow manager to link together components in a system with standardised interfaces. Paoli and Sosio (1996) adopted adapters to augment components via translating the interactions among components through inter-layer messages.

The concepts of the business process modelling techniques (e.g. Architecture of Integrated Information Systems (ARIS) (Scheer, 2000), Integration Definition for Function Modeling (IDEF0, 1993) standard, Unified Modeling Language (UML) (Rumbaugh et al., 1999), etc.) derived from structured business computing have been widely utilised in process management of CSCW applications. However, compared to the distributed system-oriented workflow management that focuses on structured processes, CSCW-oriented workflow control focuses on the unstructured processing on the shared document by human collaborators. For example, collaborators with different ownerships possess different controls over the processes (Yen et al., 2003). Therefore, a number of CSCW-oriented workflow systems automated the coordination and interoperation of workplace activities (Dori et al., 2004). Here, we just name a few. Business Process Models (BPM) (Kazanis and Ginige, 2002) is a collaborative business process modeling tool. OntoEdit (Sure et al., 2002) supports a concurrent collaborative software engineering process. Yen et al. (2003) presented a collaborative control design tool that allows privileged collaborators to change the process as needed. OPCATeam (Dori et al., 2004) integrated the object- and process-oriented paradigms into one single framework, so that structured processes and human interaction behaviours co-exist in one business process modelling system.

However, the current workflow control still has to be predefined and remain immutable. With the rapid emergence of web services technology (Alonso et al., 2004) that emphasises loosely coupled and dynamic discovery and invocation of web components support, process control needs to offer more flexibility. Therefore, we predict that the future trend will be driven by demands. The new types of control should be customisable and modifiable dynamically at the run-time.

4.2.5 Awareness control

Awareness is a fundamental and critical feature of collaborative work (Begole et al., 1997; Cadiz et al., 2002; Dourish and Bellotti, 1992; Hill and Gutwin, 2003; Mark et al., 1996) that facilitates a collaborative process (Kouzes et al., 1996). Awareness implies that a collaborator needs to be aware of actions and the progress of other collaborators (Begole et al., 1997; Convertino et al., 2004). In a word, awareness is a mechanism that keeps users aware of the states of a collaborative work.

There were few discussions in the past due to the lack of technical support on system-wide awareness control (Convertino et al., 2004; Hill and Gutwin, 2003). The only informally conceived approach is informal awareness of collaborator's activity through passive mutual monitoring (Fisher and Dourish, 2004). With the rapid development of internet and telecommunication technologies, real-time collaboration has become a reality. Therefore, researchers extensively regard awareness as essential to the success of a collaboration system. Instant messaging service is a popular text-based interpersonal awareness service that notifies people of the online presences of other people included in their lists (Greenberg and Rounding, 2001). Casca (Edwards et al., 2002) provided a similar awareness facility that detects when collaborators come online. Gräther and Prinz presented social awareness in a web-wide virtual community. Cockpit (Gräther and Prinz, 2001) provides two levels of awareness: presence of other people and personalised awareness about changes of the web sites. Notification Collage (NC) (Greenberg and Rounding, 2001) established a real-time shared surface where collaborators post multimedia elements, thus creating a rich resource for awareness. As every one in the collaboration can overhear the NC, Greenberg and Rounding found that people actively post on NC to facilitate awareness. Due to the fact that most awareness interfaces are limited to research prototypes, Sideshow (Cadiz et al., 2002) provided a Microsoft company-wide-adopted peripheral awareness interface.

Collaboration systems normally provide some system support to keep participators aware of other collaborators' status. Till date, awareness control has already been enhanced to the extent that the granularity of awareness is customisable rather than forced (Hill et al., 1994), which means that users can control the degree of awareness that they want others to know about their own work.

By granularity of awareness, we mean the part of work that a user would like to share with other collaborators. There may be a conflict between awareness and privacy (Hill et al., 1994); therefore, potentially sensitive information might need to be masked off (Boyle et al., 2000). Intermezzo (Edwards, 1995) is a framework for awareness management, where a carefully designed and fine-grained set of awareness attributes enables each user to delineate personal awareness control; and an interpretive language is provided to dynamically control the accesses based on the awareness objects associated with each user. Boyle et al. (2000) experimented on a video-based media space for various levels of awareness, and concluded that less awareness safeguards higher privacy. The MAUI toolkit (Hill and Gutwin, 2003) enabled graphical widgets to become groupware-specific components, by attaching to the widgets functionality of collecting, distributing and visualising group awareness information. BRIDGE (Ganoe et al., 2003) created awareness information through routine document transactions, and integrated the presentation of awareness information as part of workspace views. All these research works focus on providing awareness services; however, Mark et al. (1996) are concerned that the amount of awareness information may result in an information overload.

In addition, whether to keep every user's view consistent throughout the collaboration is a dilemma (Li et al., 2004). There exist two options and each has its own advantages. One option is to always distribute to all participants every change of the shared artefact. The other option is to only send modifications to a participant on-demand (Bernier, 2001). The first option is easy to implement and will keep every user's view consistent; however, it would also occupy some network traffic because of the amount of data sent to every user. The second option has the benefits of relieving network traffics, as it may save significant efforts of unnecessary distributions; however, it has to endure inconsistent views among users from time to time. Furthermore, it may require more efforts on the implementation.

Another level of awareness intends to facilitate latecomers' comprehension of the status of the collaboration (Shen et al., 2002). An example at hand is meeting minutes constructed as a meeting log (Robert III et al., 2000). Another example is TeamSpace (Geyer et al., 2001), which collects all time-based collaborative activities occurring during the collaboration, together with formal structured articulation information. Technically, having a latecomer join in a collaboration merely requires to create a new instance of the front-end application. However, latecomers normally would like to know what has happened before they join in. This information can be delivered to latecomers through two channels: one is to let the back-end application automatically send the logs of all previous activities to the latecomer, whereas the alternative is to wait until the latecomer actively queries the back-end for the information. Both approaches have their advantages. The first solution can bring the latecomers' views up-to-date, and the latter solution seems to be more efficient, since collaboration logs are likely to maintain a lot of useless information. The choice between the two options normally depends on the developers' concerns.

In addition to the above two strategies, there is a third approach that is a composition of both of them: the server only sends necessary information in the activity log to the latecomers, instead of the entire activity log. Although this approach requires more implementation on the server design, it provides a balance between efficiency and up-to-date view.

From the above analysis, we can see that the current awareness research spans broad concerns about building system-level efficient and fine-grained awareness support. We believe that future research will address the possibility of defining and updating the granularity of awareness dynamically on-demand.

4.3 Environment

Although people yearn for establishing a ubiquitous environment for collaborators to coordinate from anywhere (Dearle, 1998; Hengartner and Steenkiste, 2004), in most cases, however, a CSCW system is still a context-dependent application. Earlier CSCW applications were limited to a predefined environment. To efficiently manage a collaboration environment and enhance reusability, a CSCW system has been expected to be deployable to multiple environments, or contexts. To realise the goal of reusability, environmental concerns are separated from the CSCW application code. Actorspace (Jamali et al., 1999) presented a reflection model to separate the environmental parameters from the application code within each agent. As a result, the definition of each agent includes not only the computational part, but also a declarative specification of the requirement attributes about its execution environment. A facilitator is associated with each agent to help modify the specification dynamically. Since the Actor (Venkatasubramanian and Carolyn, 1995) model is able to directly model the computational states, Venkatasubramanian and Carolyn (1995) claimed that a computational environment can be modelled by a meta-level architecture at an appropriate level of abstraction.

Another level of environmental considerations focuses on managing the relationships between task interactions. Tasks in a real-life environment can seldom be isolated. Instead, a task is often entangled with other tasks; for example, it requires the support from other tasks. The relationships among tasks are called interdependencies (Bogia et al., 1993). Although interdependencies are usually unpredictable, they need to be appropriately managed to maintain the quality of the whole system. How to support dynamic interdependencies among collaborative activities remains a challenge. To efficiently manage the interdependencies, knowing their nature is the first step. Active (Bogia et al., 1993) aided collaborators by providing support based upon some knowledge or 'understanding' of the activities. These facilities are classified into two categories: data support facilities and process support facilities. Another research direction focuses on resource control and management.

Hariri and Mutlu (1995) adopted the Markovian technique to model component availability. Similar techniques are also utilised to perform the analysis of dependency failures among the resources and other performance constraints, such as the constraint-oriented cooperative scheduling reported by Esquirol and Lopez (1997).

These works facilitate some form of resource allocation; however, most CSCW systems pay little attention to the resource changes in a collaboration process due to environmental changes (Kosoresow and Kaiser, 1998). Thus, Begole et al. (2001) presented a semi-replicated architecture to support resource sharing in synchronous groupware. Shared artefact actually resides in a single location, while accesses from all different environments are directed through replicated proxies.

With the growing reusability and component engineering, collaborations should not be constrained to one specific environment any longer. We believe that an ideal CSCW application should be ubiquitous for different environmental contexts. One way of achieving this goal is to develop a generic descriptive formalism suitable for instantiation in multiple contexts. We consider that resource management should be the central research topic in this field.

4.4 Perspective

Cooperative work by nature is "regarded as benefiting from the combination of different perspectives, specializations, and talents" (Mark, 1997). Further, in collaborative applications, complicated semantic relationships can emerge unexpectedly so that new perspectives may appear dynamically (Li and Patrao, 2001). If we consider the problem from these various aspects, we could avoid fixing on a particular solution and bias in reasoning about relationships.

Earlier research considered perspectives centred around a single task (Lee et al., 1996). More recent work has turned to considering a collaboration as a whole. Thus, diverse research efforts focused on constructing comprehensive CSCW systems to analyse their perspectives. CBE (Lee et al., 1996) provided a shared workspace as an open environment for collaboration work with an unstructured nature. Plale et al. (1998) presented another example through distributed laboratories.

Aoyama (1998) introduced the term *agility* into software development to specify not only quick delivery of software products but also quick adaptation to changing requirements. It indicates that an approach is necessary to capture dynamically changing perspectives and reflect them to the system on-the-fly. Till date, however, the dynamics of cooperative work such as object evolution and migration, dynamic conceptual clustering and multiple perspectives/representations have not been explicitly addressed in CSCW (Bardram, 1998; Li and Lochovsky, 1998).

The essential issue for dynamic perspectives is how to capture interactions in collaboration work. By interaction we refer to an action that is influenced by the presence or the activities of other collaborators (Neuwirth et al., 1994). Interaction is fundamental in CSCW because it enables collaborators to combine their efforts effectively (Bond and Gasser, 1988; Neuwirth et al., 1994). A behaviour specification, the formal description of what is supposed to happen when software executes (Della et al., 1999) is often adopted to define an interaction. However, there is more need to be considered in addition to the operational perspective of an interaction. A language is needed for the communication process as well (Lander, 1997).

Neuwirth et al. (1994) defined a set of interaction parameters for collaborative writers. Their paper outlines the task management parameters, notification parameters and parameters for describing the scenarios of execution. Jeffay et al. (1992) specified changes to an architecture in response to run-time interactions. Astley presented a meta-architecture to help define architectural policies and facilitate the policy composition (Astley and Agha, 1998). Suite (Berlage and Genau, 1993; Dewan and Shen, 1998) system allows different degrees of coupling between individuals by associating the coupling attributes with shared objects. The coupling attributes determine which perspectives are coupled and when updates need to be propagated (Berlage and Genau, 1993). Rendezvous (Hill et al., 1994) is an object-oriented programming language, which is based on a set of constraints to ensure the consistency between multiple views and their underlying data. Spider (Boland et al., 1992) improved sharing perspectives by enriching communications among managers for distributed decision making.

Bardram (1998) further investigated the nature of CSCW on the basis of the Activity Theory (AT) (Kuutti, 1991). Three basic levels of collaborative activity were identified – coordinated, cooperative and co-constructive collaborative activities, along with the dynamic transitions between them. This paper also proposed two approaches to support dynamic transitions. One is to integrate a talk channel into CSCW systems to support transitions from the coordinated level to the cooperative level; and the other is to integrate a discussion method into CSCW systems to support transitions from the cooperative level to the co-construction level.

From the analysis, we can see that the current research on perspectives focuses on dealing with dynamic, unpredictable perspectives. Any collaboration work is based on some common rules. When rules change, the perspectives of the collaboration will change accordingly. To facilitate the reusability of collaboration frameworks for various CSCW applications, we believe that the future trend in this direction is to render per-policy-based systems. Each policy governs multiple business processes conformed to the policy; therefore, collaboration systems can be reused in a range of business process applications ruled by a common policy. How to construct a policy to represent a range of collaboration processes requires two new research directions to join into this field: HCI and social psychology.

4.5 Community

Earlier collaborations were usually performed within an organisation via intranet. Things have been changed dramatically. With the rapid acceptance of internet,

collaborations often happen among different organisations and institutions, whose culture or history may be diverse (Shen et al., 2002). In recent years, a new organisational form of virtual team emerges to group organisations into a community to achieve unprecedented levels of flexibility and responsiveness (Powell et al., 2004). Different communities need to cooperate to achieve a mutual business goal; therefore, inter- and intra-community communications become central issues for business collaborations.

AT (Kuutti, 1991) is thus a social psychological theory that focuses on the development transformations and dynamics in collective organisational activities. AT provides a set of concepts to help analyse cooperative work, especially its dynamic transformation (Bardram, 1998). The relationship between an individual's activities and other members is subject to the work division and is governed by some rules and norms (Engestrom et al., 1997).

Human communication is the primary way in which collaborative activities are accomplished between social communities. This is the reason why the computer-based communication form (such as an e-mail and conference system) is the most successful category of CSCW application till date (Bardram, 1998; Fuentes and Troya, 1999). Various types of electronic conferencing systems are built with different focuses. For example, *Talking in Circles* (Rodenstein and Donath, 2000) is an audio-conferencing system that establishes speech as the primary communication channel and supports group interaction behaviours. In the multimedia networking community, separate communication channels are usually assigned to deliver different types of data and control information that have different requirements of service quality. Actions from different communication channels thus need to be coordinated. Singh (1998) utilises communicative acts to model the exchange of information between different channels. All acts are classified into seven sets: *assertive* to inform activities, *directive* to request performance, *commissive* to slate communications, *permissive* that gives permission for an act, *prohibitive* that bans some acts, *declarative* that causes events in themselves and *expressive* that expresses emotions and evaluations.

With the increasing scale of collaboration, more and more communities may emerge. In addition, communities participating in a collaboration process may not even be evident at the beginning. Further, communities may evolve in the collaboration processes. How to deal with dynamic community evolution (such as a flash community) will remain a challenge. In addition, two other disciplines, sociology and psychology, need to be explored to properly direct communities.

4.6 Coordination

The necessity of evolving control mechanisms in CSCW is recognised and emphasised in Cortes and Mishra (1996). A coordination program defines the control mechanisms that govern an ongoing collaborative session, by constraining the methods and the way that users share the artefacts. Coordination affects the success or failure of a CSCW application, as it is the glue that binds separate activities into an ensemble (Gelernter and Carriero, 1992). The quality of the coordination achieved relies on the method to realise the coordination (Swarts, 2004).

In earlier CSCW systems, coordination was tightly coupled with computation. In recent years, coupling has been relaxed to facilitate individual development efforts for the computational and coordination parts. Further, replacement of one module will not affect other modules. In other words, the coordination policies are recommended to be separated from the computational part, which is called loose-style support of collaboration (Sato and Murakami, 1993). By separation, the policies can be specified in a more declarative way and interpreted on-the-fly (Li and Muntz, 1998). In addition, coordination policies and computational components can be developed and modified independently; therefore, information hiding, modularity and reusability can be granted.

MultiTel (Fuentes and Troya, 1999) separated communication from the data-processing components of web-based multimedia cooperative applications. Sato and Murakami (1993) investigated large-scale software collaboration and present a framework of separating management unit from information flow. Furuta and Trellis adopted a variant of coloured Petri nets to separate the implementation of coordination policies from other computational components and specified coordination policies in a declarative way (Furuta and Stott, 1994). Linda system (Gelernter and Carriero, 1992) allowed programmers to decouple the computational and coordination specifications of a parallel program. Artefact (Brandenburg et al., 1998) further decoupled input and output to enhance the support for synchronous collaboration. This level of decoupling separates the business logic for different purposes: business logic for updating the appearance of the application object, and that for sending the appearance to screens. Therefore, a shared object can be separately programmed through the three relative independent parts: computation, input and output. Cortes and Mishra (1996) conceptually split coordination mechanisms into two main categories: data mechanisms and user mechanisms. The Clover (Laurillau and Nigay, 2002) system established a coordination layer separated from communication layer.

There are also a number of discussions about whether a computation model and a coordination model should be integrated into one single language or separated and represented into two distinct languages (Gelernter and Carriero, 1992). As an example for the second option, a declarative language DCWPL was tailored to specify the coordination policies, while a language interpreter can interpret the customised coordination policies at run-time. The Cooperative Applications (Cova) (Yang, 2002) language is another example that separates coordination from computation. We believe that there are three prominent advantages for the second alternative:

1 no change to the original languages

2 no change to the platform and

3 component reusability.

Another research direction in this area focuses on how to improve the coordination between collaborators. Fussell et al. (2000) studied collaborative repair tasks and concluded that a shared visual workspace could facilitate distributed collaborations. The SPeCS system (Medeiros et al., 2001) divided the coordination process into three categories: the process management, the conflict resolution and the rationale capture. By analysing visualisations of awareness histories, Begole et al. (2002) studied the working rhythms among distributed collaborative groups to identify patterns of human activities.

However, currently the coordination policies are usually built on top of a coarse-grained service policy. In the future, we believe that coordination in collaboration can be specified at a finer-grained level and will provide more flexibility. Furthermore, with the development of component engineering, we believe that, the coordination and computation can be integrated into one component dynamically, and the integration should be able to be achieved on a per-service basis. Two important research areas in this field are middleware and coordination paradigm.

4.7 Social conduct

Computer-aided Virtual Reality (VR) was constructed to reflect complicated social conduct in collaboration work. Participants in collaboration will naturally form various groups or colonies according to certain commonality (Powell et al., 2004). Conducts within a colony will then be constrained by agreed-upon rules. Colonies may be further organised into federations of colonies of varying cultures and morals.

Professional meetings, as typical forms of CSCW activity, have been playing an increasingly integral role in our world. With the rapid development of computer and communication technologies, traditional concerns about electronic professional meetings, such as information storage and real-time communication are mostly eliminated. However, there is still very little research on a persistent question of understanding the nature of cooperative work involved with diversity of people (Bardram, 1998; Fisher and Dourish, 2004; Hovav and Mandviwalla, 1998; Millen and Patterson, 2002).

CSCW supports social interactions among groups of people; therefore, it must take into account the fuzzy concerns relating to social practices, assumptions, behaviours of people (Fisher and Dourish, 2004; Hearst, 1999) and social engagement (Millen and Patterson, 2002). Interdisciplinary work is among the most fundamental issues in CSCW research (Shapiro, 1994). Collaboration system developers must consider the psychosocial issues such as autonomy, trust, sense of place and attention to ritual. A collaboration environment is in fact similar to a square of the Information Age (Kouzes et al., 1996). Millen and Patterson (2002) explored three ways to stimulate social engagements in online communities, namely, conversation channelling and event notification, community member selections and discussion topics facilitation. Fisher and Dourish (2004) proposed to expose people's 'social workspace', the structure of the collaborative interactions that consists of social structures and temporal structures.

To make a collaboration environment productive other than merely a chat room, it is necessary to enforce some social restraints to keep a meeting in order. A variety of research has been conducted on electronic conference control over the years. Nevertheless, widely adopted robust and scalable standards are still lacking. Currently, there is still no application-independent conference control mechanism (Koskelainen et al., 2002). Padhye and Kurose (1999) defined a characterisation of observed user behaviours and provide some guidelines for designing continuous-media architectures and applications. Bormann et al. (2001) defined a Simple Conference Control Protocol (SCCP) framework together with SCCP tasks: conference management, application session management and floor control. Koskelainen et al. (2002) considered the conference management and floor control as the two main conference control components and presented a scalable conference control framework. However, all these works follow the same definition of conference control from (Bormann et al., 2001); therefore, neither of them possesses comprehensive meeting control policies, such as decision-making strategy and concurrency control. Thus, Johnson et al. (2003) believe that the understanding of the fundamental characteristics of collaborative activities is in demand.

In summary, participants in collaboration will naturally form various groups or colonies according to certain commonality. Conducts within a colony will then be constrained by agreed-upon rules. Colonies may be further organised into federations of colonies of varying cultures and morals. The main difference between a collaboration colony or federation and its political counterpart is that a CSCW process participant may belong to more than one colony and organisation. Two disciplines, sociology and psychology, will help construct effective social conduct.

4.8 Human proxy

Utilising the agent technology in CSCW research is a novel topic. Agent-based framework constructs an interactive system as a collection of specialised computational units called agents (Calvary et al., 1997). In a broad sense, an agent is any program that acts on behalf of a human user (Jamali et al., 1999; Karnik and Tripathi, 1998b). From the perspective of being an autonomous component that can take charge of some responsibilities, an agent can be considered as a human proxy, and is naturally a representative of human collaborators in a CSCW system.

An agent normally has a state, possesses an expertise and is capable of invoking and responding to events (Calvary et al., 1997). In addition, the agent style provides the capacity for software designers to allocate functionality to each agent at the appropriate level of abstraction, which is similar to the allocation of work to each person in the real world. Therefore, an interactive system can be modelled in a homogeneous way: all functional components of the system are expressed based on a single-user style (Calvary et al., 1997). Wainer and

Braga (2001) introduced into CSCW the concept of social agent, aiming at improving the quality of collaboration work. Their Symgroup system is a discussion system that embodies social knowledge by means of several social agents.

Generally, research on agent technology in the CSCW area can be classified into three directions:

1 formalised agent frameworks and languages

2 mobile agents and

3 intelligent agents.

The first direction aims at formalising the framework of agent architecture. As we mentioned above, any program set can form an agent. Facilitating the construction of an agent and the interactions between agents is important to increase reusability, extensibility and reliability of preconstructed agents. Researchers have already built patterns for the internal architecture of an agent (Jamali et al., 1999). Architecture of agent-as-building-block system, with system-supported inter-link and communication and security is a new and challenging topic. Agent Communication Language (ACL) opens another research area for the formalisation of agent technology. A set of well-defined ACLs has been developed; and some are for commercial use. Knowledge Query and Management Language (KQML, 1993; Labrou and Finin, 1999) is a pioneer; and IBM's Aglets (Karjoth et al., 1997) and ObjectSpace's Voyager (Objectspace, 1997) are typical commercial products. Jouppi (2002) adopted a tele-operated robotic surrogate to visit remote sites. Other examples can be found in Concordia (Wong et al., 1997) and Ajanta (Karnik and Tripathi, 1998a).

With the fast development of internet technology, a new term *mobile agent* has come into reality. To enable an agent to execute on heterogeneous machines with various underlying operating system environments, the portability of agent code is a prime requirement. A mobile agent is a program that represents a user in a computer network and can migrate autonomously from site to site, and to perform some computations on behalf of its dispatcher (Karnik and Tripathi, 1998b; Singh, 1998; White, 1995). To support the mobility of agent code, most agent systems are based upon interpreted programming languages that provide portable virtual machines for executing the agent code (Karnik and Tripathi, 1998b; Objectspace, 1997).

As an agent is in fact a human proxy, it is necessary for it to have some intelligence. An agent should possess the ability to make decisions when facing environmental changes, decide travel paths on-the-fly and search for target locations. Generally speaking, current research in this area is derived from AI research, especially from machine learning and artificial reasoning. AI languages, such as Prolog are widely adopted to represent knowledge and query specifications.

In summary, agents are capable of moving among hosts and making decisions autonomously with the aid of their own intelligence. However, till date the intelligence that agents possess is still limited. The degree of tolerance of decision making and the self-correcting actions taken by agents are still poorly understood. These are the critical reasons why the agent technology has not been widely used in CSCW applications. We predict that, with continued research on the agent technology, agents will become perpetual self-corrected true proxies in place of human beings. AI and intelligent agents will remain the two main research topics in the field.

4.9 Policy

A difficult issue in the development of CSCW applications is an effective and flexible specification for control and coordination (Cortes and Mishra, 1996). Similar to real life collaboration, order is essential to keep electronic collaboration valid and productive.

In the real world, some social rules are set up for participants as social protocols (Ellis et al., 1991). These rules state the rights of different users according to the roles they play within the group (Paoli and Sosio, 1996). Although it is difficult to have well-accepted coordination rules for all kinds of collaborations, we do have a set of tested and well-known rules in the real world to serve as a template, which are called Robert's Rules of Order (RRO) (Robert III et al., 2000). In the CSCW realm, similarly, interaction rules are built into the software to instruct all participants to abide by. However, to support group collaboration more effectively with the help of networked computer systems, these social policies are too complicated (Chang et al., 1999). For example, some modifications are necessary for RRO to fit CSCW environments (Zhang et al., 2003).

To make the rules implemental, these policies must be translated into machine-readable control rules, which are called coordination policies (Li and Muntz, 1998). Unlike traditional computer systems in which interaction patterns are more predictable and can be governed by hard-coded mechanical rules, it is usually not practical to hard-wire coordination policies beforehand (Li and Muntz, 1998). The definition of coordination policies depends on several factors, such as cultural background (Watson et al., 1994), the number of participants, their shared resources and user tasks or functions (Cortes and Mishra, 1996). Moreover, coordination policies are often sensitive to the work style and organisational structure of individual groups. Furthermore, coordination policies vary from collaboration to collaboration and even vary in different phases of the same collaboration (Li and Muntz, 1998), depending on the collaboration's current scenario (Cortes and Mishra, 1996). As a result, it is essential for collaboration systems to have flexibility and extensibility to allow for changes of the coordination policies at run-time.

Traditionally, if some policies are constrained into the computer-related applications, they are also known as protocols. As Bogia et al. (1993) stated, a protocol specifies a family of actions, the temporal relations among the actions and the semantics of each action. A variety of research work has been directed at mapping the flexible social protocols onto their own practical communication protocols (Neuwirth et al., 1994), such as TCP/IP, X.25 and OSI/ISO. Earlier CSCW systems predefined and hard-coded policies. After the system was delivered to the customers, policies could not be modified.

In recent CSCW applications, developers use meta-code to design patterns for policies, so that users can customise the policies at run-time. COCA technique modelled coordination policies using a Prolog-like language (Li and Muntz, 1998, 2000). It predefines different roles and explicitly defines associated coordination policies for each role. The participants are assigned to roles, and one participant can be assigned to several roles simultaneously, while several participants may also be assigned to the same role. Coordination policies are interpreted at run-time through the COCA virtual machine. The system allows participants to dynamically join or leave the collaboration by simply taking or dropping roles, and supports the dynamic modification of coordination policies at run-time. Intermezzo system (Edwards, 1995) also adopted the concepts of roles and policies, but their roles are limited to the definition of access control groups. Malone and Crowston (1990) discussed the importance of using conventions in a cooperative work arrangement. Their paper further discussed how to coordinate between heterogeneous groups using multiple perspectives to become a new convention.

Moses (Minsky and Ungureanu, 1997) utilised a controller mechanism to manage the interactions between agents in the shared space. The controller has a set of rules predefined. When a new agent is created, a new instance of control is initialised and attached to the agent. When a message is sent by an agent or arrives at an agent, this message is always forwarded to its attached controller. On the basis of the rule definition, the controller will decide whether to deliver the message to other agents or forward the message to the shared space or block the agent temporarily. In the whole system, all participants are governed by the same set of rules.

Actor-based framework (to be discussed later) models the system architecture utilising two kinds of nodes: actors and connectors. Actors are components encapsulating internal details, whereas connectors are the conjunctions between actors. Connectors can be considered as entities encapsulating all necessary coordination policies for communications among actors. Sturman (1996) modelled the interaction policies between actors in a modular manner. Astley and Agha (1998) went further to utilise a meta-architecture to model the interaction policies between actors.

In professional collaborations, a mechanism is necessary to ensure that participants can access shared data in a mutually exclusive fashion. For example, only one person who holds the floor is allowed to speak in a conference. Different collaborations require different floor control (Robert III et al., 2000) policies. In a formal situation, participants need to explicitly request the floor to talk (Hill et al., 1994). In many cases, participants should compete for the floor. Therefore, there are two requirements that a professional collaboration needs to fulfil: the ability to institute floor control and a mechanism for selecting the speaker. On the other hand, in an informal meeting, participants may be free to speak without explicit control if so desired.

With all these various policy techniques, however, these meta-codes can only model a limited range of policies. Cognitive policy patterns cannot be modelled at present, and the flexibility remains a challenge. In the future, we predict that cognitive policy patterns will be modelled, and more flexibility will be granted to model an extensive range of policies. In addition, we believe that the decision support and the extensions to traditional RRO (Robert III et al., 2000) to suit formal electronic meetings will be the two main research areas in the field.

4.10 Storage schemas

Storage issue has always been one of the major issues in distributed collaborative computing, which requires a strong consistency for shared persistent data and efficient access to fine-grained objects (Bjornsson and Shrira, 2002). There are two approaches of distributing shared information in a CSCW system: display broadcasting and event broadcasting (Begole et al., 1997). The display broadcasting approach aims at distributing graphical depiction of the shared application to each participant's display screen; whereas the event broadcasting approach aims at distributing only application events. Each approach has its advantages and disadvantages. There is much less data that needs to be distributed in the latter approach; therefore, the communication overload can be largely reduced. Meanwhile, platform independence and mobility of the system are also promising. However, event broadcasting requires that the application be duplicated at every client site; therefore, it leads to a heavyweight application. Further, platform compatibility needs to be considered, as each client will generate her own display based on the events received. Through display broadcasting, no duplication of application leads to lightweight clients. However, distributing display information can easily cause a traffic bottleneck in the system.

There are other approaches that intend to reduce the burden of information distribution in a CSCW system. Berlage and Genau (1993) built a tree-like structure to store *command* objects. By only storing common cases once, the transmitted information can be largely reduced. In addition, this structure easies and shortens the definition of specific features of an application. Utilising spatial metaphors to represent information and action in electronic systems is presented in Trevor et al. (1998). Their work is rooted in the use of a 'room'-based metaphor to allow the presentation of information. The UARC system also utilised a room as a metaphor to support both synchronous and asynchronous collaboration (Subramanian and Malan, 1999). The CAOS system used an incremental spatial parsing mechanism to store both information and the parsed spatial structure (Reinert et al., 1999). For efficiency reason, the parser is incremental, so that only the regions that are actually affected by changes need to be reparsed. Schulz et al. (1998) used simulation-based design to assess virtual prototypes before a system is actually constructed. BuddyCache (Bjornsson and Shrira, 2002) is a transactional caching approach that improves

the latency of access to shared persistent objects in high-latency network environments.

With the development of mobile agents and component engineering technologies, we believe that in the future, the storage issue can be eventually solved by mobile components and their self-destruction after their jobs are finished. These mobile components will eventually become information cells that possess ability of auto-presence, self-purge and self-migration. Research areas in this field will focus on object-oriented databases, multi-tier clients and server architecture.

4.11 Information presentation

Information presentation in CSCW usually adopts the style of hypermedia. Hypermedia plays two central roles in the CSCW domain: providing a medium for both content and cooperation/coordination information (Mark et al., 1996). Mark et al. utilised hypermedia in CSCW systems to facilitate the division of tasks and labour. A highly modular structure of shared document is presented to divide the work among the group members to create modules in parallel and to integrate them at a later stage.

Metadoc (Boyle and Feh, 1993) attempted to vary the content and the details of information presented to a reader, according to the reader's ability and the requirements. IBM's Lotus Notes (Logrippo et al., 1992) is one of the most outstanding works for generating multimedia documents. It provides a general pattern for users to describe the multimedia information and it also provides system support to integrate different kinds of information together in a *RichDocument* format.

Due to the emergence of World Wide Web, associative structure or linking is currently the most popular hypermedia structuring mechanism; and link traversal is the most popular hypermedia navigation mechanism (Will and Hicks, 2000). Fluid Open Hypermedia prototype (Bouvin et al., 2002) integrated with the web standards to support fluid annotations to facilitate collaborative editing and augment personalised data mining over the internet. This system provides a fine-grained annotation within a page, ranging from any character or object on the page, to existing linking anchors.

With the development of graphics technology, multimedia research and hardware support, immersed VR promises an amazing future for efficient and effective collaboration applications. We believe that graphics, HCI, VR, and tele-immersion will be the central research areas in the field.

4.12 Formalisation

In general, the greater the level of formalisation required, the greater the level of functionality needed (Shipman and Mccall, 1994). Therefore, there have been a number of researchers focusing on the formalisation of CSCW systems, either on a formal framework or on a tailored formal language.

4.12.1 Architecture

Software architecture plays an essential role in software systems by providing the plausible insights, triggering the right questions and offering general tools for thoughts (Calvary et al., 1997). An architecture is a semi-complete representation that contains certain fixed aspects common to all similar applications in the problem domain, along with certain variable aspects unique to each specific application (Srinivasan, 1999). Building 'conceptual constructs' is fundamental not only for a single software design, but also for all large-scale applications (Jeffay et al., 1992).

In the CSCW realm, there are three general architectural models derived from distributed computing:

1 client/server model

2 actor-based model and

3 layered architecture.

The *client/server model*, ranging from the basic two-tier model to three-tier to multi-tier model (Robert et al., 1999), is still the dominant framework utilised in the CSCW world. In recent years, the middleware concept and its related commercial products, such as OMG's CORBA (1997), Microsoft's DCOM (Rubin and Brain, 1999) and Sun's EJB (Thomas, 1997), inject into the traditional client/server model more modularity, reusability, extensibility and reliability.

VCE is a client/server-based virtual conferencing framework that relies on conference servers to facilitate audio handling (Prasad et al., 2003). The 'generic multi-user architecture' model (Dewan and Shen, 1998) structures a Groupware system into a variable number of levels of abstraction, ranging from the domain-specific level to the hardware level. In this model, the number of the functional layers is left open. Hariri and Mutlu (1995) presented a two-level hierarchical model to analyse the availability of distributed collaboration systems. The user level is at the higher level to analyse the availability of the tasks (processes) using a graph-based approach; and the component level is at the lower level to analyse the component availability using detailed Markov models. Calvary et al. (1997) modelled the information flow between layers through vertical and horizontal directions. Through the vertical direction, information flows between adjacent layers along the input and output axes; through the horizontal direction, information flows between peers or non-peer replicated layers for state synchronisation. Another system, EGRET, also includes a simple structural inheritance mechanism to create hierarchies of schemas (Johnson, 1992).

Following the boom of Microsoft Distributed Object Computing (DOC), a*ctor-based model*, as an object-based framework, emerges to become a new design paradigm for distributed collaboration systems. Implementation details are encapsulated inside actors; resource constraints are associated with actors and communication and coordination are realised through asynchronous message passing between actors. Jamali et al. (1999) presented an actor-based architecture for customising and controlling

agent ensembles. Trevor et al. (1998) presented an extensible session architecture to support the management of heterogeneous virtual landscapes, whose model is based on three system-provided services: user profile service, registry service and personal service. Sato and Murakami (1993) stored computational information and relationship separately. Computational information is stored in nodes, whereas relationships between nodes are represented by links. Yang (2002) proposed a uniform meta-model that consists of a set of collaboration activities.

A variant of the actor-based model is a role-assigned model, where actors are further categorised by roles. The group of actors who are governed by the same set of coordination policies will act the same role. Ellis et al. (1991) defined user roles as a set of privileges assigned to a group of users. Multitel system (Fuentes and Troya, 1999) described a service architecture that defined the user interaction logic for specific services. Three kinds of actors are identified: an *organiser* is the participant who calls for the cooperation; a *participant* is one who can receive video and audio from the service provider or from other participants; a *manager* is a special participant who is in charge of starting and managing an organised service. In the ADOME system (Li and Lochovsky, 1998), the roles define the social commitments or obligations of members in a group. The commitments are restrictions on how members must act. When a member joins a group, she chooses one or more roles, thereby acts according to the commitments associated with the roles. Hawryszkiewycz (2001) presented a meta-model structure, in which the central component is a *workspace* where activities take place. A workspace is composed of a set of concepts; and each concept is defined as a role that is assigned responsibilities within the workspace.

The third popular model is the well-known *layered architecture*. Clear boundaries between the adjourning layers and the support from lower layers facilitate the design and implementation of a system, and also encapsulate the development details of each layer from its adjacent higher layer. The MOVE collaboration environment (García et al., 2002) presented a component groupware framework called ANTS. Three layers are defined in the system: a technology layer, a CSCW service layer and an application layer. Every layer possesses a component model that incorporates system-supported services. Vin and Chen (1992) built a three-layer model. The streams layer handles the media communication modulated by access rights; the sessions layer represents the collections of semantically related media streams; and the conferences layer manages temporally related sequences of sessions. The Clover (Laurillau and Nigay, 2002) system also defined a three-layer structure: production, coordination and communication.

Layered systems can be enhanced in the sense of modularity, if inter-layer communication is handled by connectors. Connectors are defined almost the same as those in the actor model, where they take the responsibility of enabling computing components to communicate, and encapsulating any distribution and network connectivity management issues (Paoli and Sosio, 1996). To improve the flexibility, Paoli and Sosio (1996) replaced the point-to-point connectors with the cooperative connectors multiplexing and demultiplexing I/O between a specific layer and multiple instances of its lower layers.

A replicated architecture is claimed to be appropriate for CSCW applications (Russel et al., 1996). In this model, applications are duplicated at each user's site and users work on the copies of the shared object on their local machines, and communications are exchanged through message passing. Individual work is done locally before it is finally merged into a product. In this way, users may continue their local work even when the network fails. This architecture indeed reflects an asynchronous CSCW style. One example application using replicated architecture can be found in DistView (Prakash and Shim, 1994).

In general, the three architectural models mentioned above are designed for general distributed computing and are weak in supporting rigorous cooperative work. There is a need to envision a generic CSCW framework with specific concerns on cooperative work, coordination policies and collaboration rules. How to combine the advantages of these three models to construct a specific CSCW-oriented framework remains a challenge.

Other research is concerned with interactions between different components in a CSCW system. For example, Berlage and Genau (1993) used a tree of command objects to model the interaction history of a shared document. Buhr (1998) presented a scenario-based notation called 'Use Case Map' to describe, in a high-level abstraction, how the architectural structure of a complex system and the use-case-aware behaviour of the system are distributed. The notation aims to help people visualise and outline the big picture of a complicated system. Roussev et al. (2000) presented an infrastructure design utilising programming patterns as a means of separating the logical structure of shared artefacts from the implementation details. An event-based model is adopted to enhance the communications between components. Dragonfly (Anderson et al., 2000) presented an architectural style that maintains a tight and bi-directional link between conceptual CSCW architecture and the implementation of CSCW architecture. Evolutions can occur at both levels independently, by the way of splitting each component into facets and each facet handles one aspect of collaboration control. Speakeasy (Edwards et al., 2002) framework is most concerned with interface design, where each component exposes its functionality via a small number of fixed interfaces that are able to interact with each other.

One essential criterion to judge an architectural model for CSCW applications is whether the model facilitates the interactions between collaborators. From the descriptions above, we can see that an actor-based model offers autonomy, flexibility and extensibility. However, as the common goal of the group needs to be fulfilled by the collaboration, different individual actor components need to support each other at different time as servers. Meanwhile, in a large-scale collaboration, hierarchical relationships may be necessary to ensure the achievement of the common interest. Therefore, we believe that more process-oriented framework would seamlessly integrate

the advantages of the three categories of architectural models. In other words, we believe that the future model will be based on context-constrained objects and it will be a loosely coupled actor-based model.

4.12.2 Architectural languages

Dedicated Architecture Description Languages (ADLs) are important techniques to formalise an architectural model, to precisely model and track the execution of a system. A variety of ADLs have been defined in the literature (Astley and Agha, 1998; Jiang et al., 2002; Li and Muntz, 1998). These ADLs were designed to capture not only the static structure and properties, but also the dynamic structure of CSCW systems (Astley and Agha, 1998). However, the unique features of the CSCW realm decide that more issues need to be addressed, such as resource management and component collaboration (Brandenburg et al., 1998).

Artefact (Brusoni et al., 1997) specified objects by defining their behaviours and appearances in an XML-like language called Artifact Definition Language. The system provides a set of base classes for developers to customise the associated access control. Later (Luckham et al., 1995) defines a three-layer query language that allows easy manipulation and query to a temporal knowledge base.

Rapide (Yang, 2002) is an object-oriented language that consists of three components with events acting as interaction descriptors. Its interfaces define a set of named entry points and connection rules determine the behaviour of the architecture by describing how events can be transmitted between interfaces. Cooperative Applications (Cova) (Li and Muntz, 2000) language is a uniform model-based ADL that separates coordination from computation of collaboration systems. Li and Muntz proposed a Prolog style description language for collaboration applications. The language also separates coordination policies from user interfaces (Allen and Garlan, 1994).

Wright (Hoare, 1998) statically defined an actor-based system architecture by extending Communicating Sequential Processes (CSP) (Astley and Agha, 1998). A component is defined by an interface with a set of ports and an encapsulated computation block that describes component behaviours. A connector defines the interaction policies between components with the support of roles and rules. The roles specify the relationships between the components in an interaction and the rules describe the low-level events between the ports for the interaction.

DCL (Della et al., 1999) defines modules and protocols based on rules, so that it can define potential architectures in response to run-time interactions. In other words, DCL architectures are reconfigurable at run-time. In addition, DCL's meta-architecture allows users to customise policies that describe interactions and resource acquisition of components and connectors.

In summary, the current CSCW ADLs focus on describing a specific architectural model or collaboration policies. It has become much more flexible to support platform- and application-independent processing. The trend, we believe, will be human-centric natural language. Users will not work with complex and detailed specification languages; instead, a set of easily understandable icons and images may help users to construct the system easily.

4.12.3 System composition interface

System composition interface is normally defined using Interface Definition Language (IDL). IDL is utilised to describe the interface of a component in a system and to enforce the communications among components through the interface (Arisha et al., 1999). Some systems provide specific languages to describe system-supported services. For example, Impact (Della et al., 1999) presented a simple but restricted service description language to formally describe the services that an agent can provide. To overcome the shortcomings of limited descriptive capabilities for attributes and operations of current IDLs, Biscotti integrated IDL with Java to specify the behaviours of components, thus facilitates the monitoring assertions at run-time using Java's reflection capabilities (Mark, 1997). The use of an IDL (e.g. CORBA's IDL) or a communication language (e.g. KQML (Lander, 1997)) provides a foundation for communication. Communication languages, such as KQML, can also be used in conjunction with special data representation languages (e.g. Knowledge Interchange Format (KIF)) (Lee, 1998; Lee et al., 2001). Once a communication language is established, the communication interoperability requires system support to ensure that it is appropriately interpreted.

In summary, these interfaces defined based on an interface language have expanded their usages from development process to the whole life cycle of the software component. It is promising that future systems can be quickly constructed from various available components based on their precise descriptions. Therefore, we predict that the future trend of system composition should be *componentware*.

4.13 Summary

We have surveyed the realm of CSCW along the 12 most significant research topics centred around the 3C model. In each topic we have analysed its past approaches and current status and predict the future trends. To help readers, we have an overall picture of our analysis: we put past achievements, current status and future trends in Table 1 for all topics.

5 Business process requirements analysis

In this section, we will discuss why and how the achievements from the CSCW area can benefit business process integration and management.

A modern business process normally involves a group of human participants. People who participate in the same business process can be distributed geographically as well as organisationally. As a result, to ensure the success of a distributed business process (Hodel et al., 2004), it is critical to enable and facilitate distributed collaborations. CSCW technology intends to provide a virtual

Table 1 Paradigm shifts in CSCW research

	Past	*Now*	*Future*	*Research areas*
Application style	Single-user/ Asynchronous	Multi-user/Synchronous	Community-based ubiquitous	Application domain analysis
Control				
Concurrency	Single-thread	Multi-thread	Massively parallel	Operating systems, Networking, Security, Privacy
Access	Linear access matrix	Hierarchical access matrix	On-demand based or customised	
Awareness	Lack of awareness	Forced awareness		
Session	Rigid/Predefined	Flexible/Dynamic		
Workflow	Linear/Centralised	Multi-dimensional/ Distributed		
Environment	Single context	Multiple contexts	Ubiquitous	Resource management
Perspective	Per task	Per collaboration	Per policy	HCI, Psychology
Community	Isolated	Cooperative	Evolutionary (flash)	Sociology, Psychology
Coordination	Tightly coupled with computation	Coarse-grained service policy, loosely coupled with computation	Fine-grained service policy with on-demand computation	Middleware coordination paradigm
Social conduct	Community-sanctified etiquette	Computer-aided virtual community *netiquette*	Federation of colonies of varying cultures and moral	Sociology, Psychology
Human proxy	Static/Mechanic	Mobile/Intelligent	Perpetual self-corrected true proxy	AI, Intelligent Agents
Policy	Predictable pattern/Hard-coded	Unpredictable pattern/Meta-coded	Cognitive/Policy patterns	Decision support, RRO
Storage schemes	Heavy-weight/light-weight (trade-offs) fat client, archive	Applet/Plug-in (alternatives) thin client, portal	Auto-presence/Self-purge self-migration, information cell	Object DB, Multi-tier client/server architecture
Information presentation	Text/Graphics/Static	Multimedia with A/V/Dynamic	VR/Immersive	Graphics, HCI, VR, Tele-immersion
Formalise				
Collaboration architecture	Function/ Structure-based	Object/Loosely coupled object-based	Context-constrained objects/Tightly coupled actor-based	Focused on Formal electronic meeting systems
Collaboration language	System-centric (programming languages)	Knowledge-centric (domain language)	Human-centric (natural language)	
System composition	Brute-force glueware	Middleware	Componentware	

collaboration technique to enable distributed collaborators to work on a common interest effectively and efficiently. The largely overlapping common intension thus makes CSCW a natural backbone to support distributed business processes. In the rest of the section we will discuss, along the 12 CSCW research directions that how the current achievements from each direction can benefit distributed business process management.

To facilitate our discussion, we will adopt a typical distributed business process – distributed software project management – as an example to examine each of the 12 directions. A distributed software project management system needs to coordinate a team of distributed software developers, testers, analysts and project managers to work on the same business software project for the project to be delivered in time. We chose a software project management environment due to its following unique complex features:

1 project management is a business process that lasts for a period of time
2 project management is a large-scale business process that contains many hierarchical smaller-scale processes, such as code check-in and project integration testing
3 project management involves a team of members acting in different roles, and it requires a lot of collaborations and interactions among members
4 project management needs to provide an environment that enables a team member to be aware of the teammates' work

5 project management requires a central resource repository and access control to the repository

6 project management requires an environment that favours both asynchronous and synchronous collaboration and

7 project management requires a comprehensive policy enforcement.

At the design phase of this project Flywheel, we identified the above challenges and turned to existing CSCW research results. We have found that many challenges can be addressed accordingly. In the rest of the section, using the commercial distributed software management tool Flywheel (http://flywheel.chicagojava.com) that utilises the present CSCW research achievements, we will illustrate how a well-designed CSCW system can fulfil the requirements of distributed business process.

5.1 Application style

Flywheel is installed on a central server machine. As a project lasting for a certain period of time, Flywheel has a central database to maintain the shared project documents, including a version-controlled project code base and all project-related documents such as analysis reports, design documents and test cases. With this central storage server, everybody in the team can be brought to the same page at any time.

Flywheel provides a virtual collaboration environment to support all team members. They can either work asynchronously on their own schedules or synchronously, for example, one day two developers work on a same code file from different machines. Every member downloads a client version of Flywheel onto his/her local machine. By synchronising with the central server, each member looks at the same version of the project.

Every team member always works on her local version first. After the work is done and tested, the updates will be merged into the Flywheel central server. Flywheel maintains the consistency of the central database by enforcing check-in and check-out rules.

5.2 Control

Project management poses requirements on all five control issues: concurrency control, access control, awareness control, session control and workflow control.

5.2.1 Concurrency control

When multiple members login to the Flywheel system, they are running on concurrent threads. When they try to access the shared database simultaneously, concurrency control is granted to maintain the consistency of the project status. For example, simultaneous writing is prohibited while simultaneous reading is allowed.

5.2.2 Access control

Different team members have different privileged rights to access each shared project file. For example, testers can only read files without updating files; and a junior developer has no access to some system-level files. In other words, access privileges (Robert III et al., 2000) are assigned to different members based on their roles in the project. A member may act as multiple roles, for example, a person may be an architect and a project manager. The access control is granted at the time of his/her loging in.

5.2.3 Awareness control

Flywheel allows each user to monitor whether other team members log into Flywheel also. Each logged in user has a view of all other members logged in; thus, she can choose to use a private messenger channel to talk to her teammates. In addition, when a team member intends to update a project file and finds that the file is being locked, she can query the system who has checked out the file; therefore, she may try to negotiate with the person for the file access.

5.2.4 Session control

The version control system in Flywheel utilises the concept of session to manage shared project source files. Each version of the source control is considered as a session of collaboration. The result of each session of collaboration is an executable version of the project. Administrators can add new team members into the session and remove members if so desired.

5.2.5 Workflow control

Multiple developers simultaneously work on the source databases. The access of a file in the source control is defined as a workflow process (check $\rightarrow$ lock $\rightarrow$ update $\rightarrow$ unlock). A user needs to first check whether the file to access is locked. If the file is locked, she has to wait; otherwise, she will lock the file, update the file and unlock the file. In addition, a user can lock a whole directory instead of a single file. At any time, there may be multiple users trying to access one same file, and there may be parallel tasks accessing at various layers in the source control. Flywheel facilitates all of these workflow control.

5.3 Environment

Project management is not constrained to one specific environment. Instead, Flywheel client version is downloadable from anywhere from the internet. As it is a Java-based application, it can run on Java Virtual Machine (JVM) at any platform.

5.4 Perspective

Considering the overall project management as the highest level of business process, Flywheel supports at different levels multiple business processes with different perspectives. A set of types of business processes are identified in the system: different types of source control, requirements documents management, brainstorming design analysis, test cases management and task

acceptance management. Each type of business process is governed by individual predefined policy.

5.5 Community

Normal project management is conducted inside an organisation; thus, there is only one community involved. However, project management can also happen among different organisations, when multiple organisations cooperate on a common project. That being the case, different communities need to cooperate to achieve a mutual business goal. Flywheel provides an administrative tool to allow various rules to be customised and predefined for different business processes dynamically.

5.6 Coordination

Flywheel provides an effective and efficient collaboration environment for team players. Team members in different roles collaborate to incrementally build the project code base in the central source control system. Flywheel provides a means for members to have real-time meetings to discuss. Meeting minutes are instantaneously generated and stored into the central server place, so that latecomers can use to be brought to the same page. In addition, an internet-messenger-alike group messenger application is provided for group members to have private conversations.

5.7 Social conduct

The term *netiquette,* initially for *etiquette* in the e-mail systems, is used in Flywheel to represent the social conduct in the constructed virtual team room. Team members joining in project collaboration engage in complicated relationships; therefore, the etiquette is used to govern participants' relationships. Conducts within a project role, for example, developer, will be constrained by agreed-upon rules. A member can act as more than one role though, for example, an architect can also serves as a part-time project manager. Thus, Flywheel adds a *role* item at the login process to represent this type of scenario.

5.8 Human proxy

Current Flywheel version does not use agents technology.

5.9 Policy

Enforcement of policies in Flywheel possesses a meta-code style. For the identified business processes in project management, the developers of Flywheel use metacode to design patterns for policies and users can customise the policies at run-time. For example, users have different options on check-out policy: one is to allow simultaneous check-outs meaning that multiple users can try to compete to check-in code; the other one is the opposite meaning that at any time, if a source code file is checked out, other users can only access the file with read-only privilege.

5.10 Storage schemas

Flywheel balances between heavy- and light-weighted clients. All of the shared information is maintained and managed at the central server. However, all users of Flywheel download and install a client version on the local machine to perform any operations locally.

As a software project management tool, Flywheel can store text files and image files. To enhance flexibility and ubiquity, all text files are stored not in traditional databases; instead, they are stored into XML (McLaughlin and Loukides, 2001) format, which is a universal data format for structured information.

5.11 Information presentation

Flywheel supports text and graphics presentation. Java Swing (Swing, 1999), a Java 2 interface that web developers use to create Graphical User Interface (GUI) is adopted to build user-friendly environment.

5.12 Formalisation

At the highest level, Flywheel utilises a client/server architectural model. A central server is set up to host all project-related information. All activities that are ready to become permanent are conducted at the central server site. Downloadable client sites provide an offline place for project members to work on local versions.

Flywheel is coded in Java and Java 2 Platform, Enterprise Edition (J2EE, 1999) technology. Java is an interface language, which facilitates design-by-interface and component composition.

6 Conclusions

This paper has surveyed and analysed the major CSCW research categories. We have used a project management tool to illustrate how CSCW research provides a technical basis to facilitate distributed business processes. We believe that distributed business process integration and management can be significantly enhanced by utilising the achievements from the CSCW field.

However, there is another problem business practitioners will encounter. Different business processes possess different business rules. Therefore, one CSCW environment built for one specific business process may not be fully reusable for various business processes. To further facilitate business process management and collaboration using CSCW, we believe that the two main directions deserve future research. The first direction is to construct a generic model for business collaboration work, such as a general architecture along with a formal language to define the properties and attributes of a system. The second direction is to research how to further improve system support for various aspects of business cooperative work, such as awareness control, flow control, coordination, etc.

As previously mentioned, all the issues discussed above should be addressed when a generic system

framework is built. We summarised these current research issues in the CSCW realm in Table 1. However, after inspecting these topics, we have found that these design issues span a very broad range of fields, as shown in Table 2. In our continuing effort to build a generic system framework, we will exclude some of the issues from consideration, which include issues such as application style, perspective, social conduct, human proxy, and multimedia/hypermedia. These topics are strongly related to other research domains. Application style is a technology trend issue; perspective is an HCI issue; social conduct is mainly a sociology issue; human proxy falls within the AI arena; and multimedia/hypermedia is mostly graphics- and HCI-related topics. Therefore, they are beyond the scope of our future research. Our research goal is to establish a generic model that is capable of supporting a large variety of business process applications; and to seamlessly integrate related methodologies to assist users to rapidly develop new business process applications. In other words, this generic model should integrate considerations and solutions to most, if not all, of the previously discussed computer science-related research topics; thus, it can support a variety of heterogeneous collaboration requirements. Therefore, the formalisation of business process applications is our research objective.

Table 2 Classification in CSCW research

Issue	*Research area*
Application style	Technology trend
Control	
Concurrency	Operating systems,
Access	Networking,
Awareness	Security,
Session	Privacy
Workflow	
Environment	Resource management parameterisation
Perspective	HCI, Psychology
Community	Sociology, Psychology
Coordination	Middleware, Compositional paradigm
Social conduct	Sociology, Psychology
Human proxy	AI, Intelligent Agents
Policy	Decision support, RRO
Storage scheme	Formalisation
Information presentation	Graphics, HCI
Formalise	
Architecture	Our main research objectives in the area of CSCW
Language	
System composition	

In the next step of our research, we intend to integrate all formalisation-related issues into the consideration of establishing a generic architectural model for business process-oriented collaboration applications. As given in Table 2, these issues include control issues, community management, coordination management, policy assignment, storage schemes, architecture, language support, interface language definition and so on. We admit that it is truly difficult to demonstrate that one model is the best for all kind of business process applications or most suitable to every category of applications. However, Tables 1 and 2 summarise a wide range of research issues that business process applications development will confront. They at least point out many, if not all, pertinent research issues to tackle towards an extremely complicated problem domain in support of business processes.

References

Allen, R. and Garlan, D. (1994) 'Formalizing architectural connection', *Proceedings of IEEE 1994 International Conference on Software Engineering (ICSE)*, pp.71–80.

Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004) *Web Services: Concepts, Architectures and Applications*, Springer Verlag.

Anderson, G.E., Graham, T.C.N. and Wright, T.N. (2000) 'Dragonfly: linking conceptual and implementation architectures of multiuser interactive systems', *Proceedings of the IEEE 22nd International Conference on Software Engineering (ICSE)*, Limerick, Ireland, pp.252–261.

Aoyama, M. (1998) 'Web-based agile software development', *IEEE Software*, November/December, pp.56–65.

Arisha, K.A., Özcan, F., Ross, R., Subrahmanian, V.S., Eiter, T. and Kraus, S. (1999) 'Impact: a platform for collaborating agents', *IEEE Intelligent Systems*, March/April, pp.64–72.

Astley, M. and Agha, G.A. (1998) 'Customization and composition of distributed objects: middleware abstractions for policy management', *Proceedings of the ACM SIGSOFT Sixth International Symposium on Foundations of Software Engineering*, pp.1–9.

Bardram, J. (1998) 'Designing for the dynamics of cooperative work activities', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.89–98.

Begole, J., Smith, R.B., Struble, C.A. and Shaffer, C.A. (2001) 'Resource sharing for replicated synchronous groupware', *IEEE/ACM Transactions on Networking (TON)*, Vol. 9, No. 6, pp.833–843.

Begole, J., Struble, C.A. and Shaffer, C.A. (1997) 'Leveraging Java applets: toward collaboration transparency in Java', *IEEE Internet Computing*, March/April, pp.57–64.

Begole, J.B., Tang, J.C., Smith, R.B. and Yankelovich, N. (2002) 'Work rhythms: analyzing visualizations of awareness histories of distributed groups', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.334–343.

Berlage, T. and Genau, A. (1993) ' A framework for shared applications with a replicated architecture', *Proceedings of the Sixth Annual ACM Symposium on User Interface Software and Technology (UIST)*, pp.249–257.

Bernier, Y. (2001) 'Latency compensating methods in client/server in-game protocol design and optimization', *Proceedings of the Game Developers Conference.*

Bjornsson, M.E. and Shrira, L. (2002) 'BuddyCache: high-performance object storage for collaborative strong-consistency applications in a WAN', *Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, Seattle, WA, pp.26–39.

Bogia, D.P., et al. (1993) 'Supporting dynamic interdependencies among collaborative activities', *Proceedings of the ACM Conference on Organizational Computing Systems (CODS)*, pp.108–118.

Boland, R.J., Maheshwarl, A.K., Te'eni, D., Schwartz, D.G. and Tenkasi, R.V. (1992) 'Sharing perspectives in distributed decision making', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.306–313.

Bond, A.H. and Gasser, L. (Eds) (1988) 'An analysis of problems and research in DAI', *Readings in Distributed Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann, pp.3–35.

Bormann, C., Kutscher, D., Ott, J. and Trossen, D. (2001) *Simple Conference Control Protocol Service Specification, Internet Draft*, WIP Internet Engineering Task Force, Editor.

Bouvin, N.O., Zellweger, P.T., Grønbæk, K. and Mackinlay, J.D. (2002) 'Fluid annotations through open hypermedia: using and extending emerging web standards', *Proceedings of the 11th International Conference on World Wide Web (WWW)*, Honolulu, HA, pp.160–171.

Boyle, C. and Feh, S.H. (1993) 'Multimedia intelligent documentation: metadoc V', *Proceedings of the ACM 11th Annual International Conference on Systems Documentation (SIGDOC)*, pp.21–27.

Boyle, M., Edwards, D. and Greenberg, S. (2000) 'The effects of filtered video on awareness and privacy', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Philadelphia, PA, pp.1–10.

Brandenburg, F., Byerly, B., Dobridge, T., Lin, J., Rajan, D. and Roscoe, T. (1998) 'Artefact: a framework for low-overhead web-based collaborative systems', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Seattle, WA, pp.189–196.

Brusoni, V., Console, L., Pernici, B. and Terenziani, P. (1997) 'Later: managing temporal information efficiently', *IEEE Expert*, July/August, pp.56–64.

Buhr, R.J.A. (1998) 'Use case maps as architectural entities for complex systems', *IEEE Transactions on Software Engineering*, Vol. 24, No. 12, pp.1131–1155.

Cadiz, J.J., Venolia, G., Jancke, G. and Gupta, A. (2002) 'Designing and deploying an information awareness interface', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.314–323.

Calvary, G., Coutaz, J. and Nigay, L. (1997) 'From single-user architectural design to PAC: a generic software architecture model for CSCW', *Proceeding of the ACM Conference on Human Factors in Computing Systems (CHI)*, Atlanta, GA, pp.242–248.

Chang, C.K., Zhang, J. and Quek, F. (1999) 'Rule-mitigated collaboration technology', *Proceedings of IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, 20–22 December, Cape Town, South Africa, pp.137–142.

Cohen, J. (1996) 'Computer medicated communication and publication productivity among faculty', *Internet Research Electronic Networking Applications and Policy*, Vol. 6, No. 23, pp.41–63.

Convertino, G., Neale, D.C., Hobby, L., Carroll, J.M. and Rosson, M.B. (2004) 'A laboratory method for studying activity awareness', *Proceedings of the ACM Third Nordic Conference on Human–Computer Interaction*, Tampere, Finland, pp.313–322.

CORBA (1997) *CORBA Services: Common Object Services Specification, Version 3*. Object Management Group.

Cortes, M. and Mishra, P. (1996) 'DCWPL: a programming language for describing collaborative work', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.21–29.

CVS (2004) 'Concurrent versions system', Available at: https://www.cvshome.org.

Dearle, A. (1998) 'Toward ubiquitous environments for mobile users', *IEEE Internet Computing*, January/February, pp.22–32.

Della, C., Cicalese, T. and Rotenstreich, S. (1999) 'Behavioral specification distributed software component interfaces', *IEEE Computer*, July, pp.46–52.

Dewan, P. and Shen, H. (1998) 'Controlling access in multiuser interfaces', *ACM Transactions on Computer–Human Interaction*, Vol. 5, No. 1, pp.34–62.

Dori, D., Beimel, D. and Toch, E. (2004) 'OPCATeam – collaborative business process modeling with OPM', *Proceedings of Second International Conference on Business Process Management (BPM)*, 17–18 June, Potsdam, Berlin, Heidelberg, Germany: Springer-Verlag, pp.66–81.

Dourish, P. and Bellotti, V. (1992) 'Awareness and coordination in shared workspaces', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 1–4 November, pp.107–114.

Du, L., Li, R., Yu, Y. and Yang, Y. (2003) 'Using familiar single-user editors for collaborative editing', *Proceedings of 36th Annual Hawaii International Conference on System Sciences (HICSS)*, January, pp.40–49.

Edwards, K. (1995) *Coordination Infrastructure in Collaborative Systems*, Atlanta, GA: Department of Computer Science, Georgia Institute of Technology.

Edwards, W.K. (1996) 'Policies and roles in collaborative applications', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.11–20.

Edwards, W.K., Newman, M.W., Sedivy, J.Z., Smith, T.F., Balfanz, D., Smetters, D.K., Wong, H.C. and Izadi, S. (2002) 'Using speakeasy for ad hoc peer-to-peer collaboration', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.256–265.

Ellis, C.A., Gibbs, S.J. and Rein, G.L. (1991) 'Groupware: some issues and experiences', *Communications of ACM*, Vol. 34, No. 1, pp.38–58.

Engestrom, Y., Broad, A.K., Christopher, L. and Gregory, J. (1997) 'Coordination, cooperation, and communication in the courts', in M. Cole, Y. Engestrom and O. Vasquez (Eds). *Mind, Culture, and Activity*, Cambridge: Cambridge University Press.

Esquirol, P. and Lopez, P. (1997) 'Constraint-oriented cooperative scheduling for aircraft manufacturing', *IEEE Expert*, January–February, pp.32–39.

Fisher, D. and Dourish, P. (2004) 'Social and temporal structures in everyday collaboration', *Proceedings of the ACM 2004 Conference on Human Factors in Computing Systems*, 24–29 April, Vienna, Austria, pp.551–558.

Fuentes, L. and Troya, J.M. (1999) 'A Java framework for web-based multimedia and collaborative applications', *IEEE Internet Computing*, March/April, pp.55–64.

Furuta, R. and Stotts, D. (1994) 'Interpreted collaboration protocols and their use in groupware prototyping', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.121–131.

Fussell, S.R., Kraut, R.E. and Siegel, J. (2000) 'Coordination of communication: effects of shared visual context on collaborative work', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Philadelphia, PA, pp.21–30.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading, MA: Addison-Wesley Publishing Co.

Ganoe, C.H., Somervell, J.P., Neale, D.C., Isenhour, P.L., Carroll, J.M., Rosson, M.B. and McCrickard, D.S. (2003) 'Classroom BRIDGE: using collaborative public and desktop timelines to support activity awareness', *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, Vancouver, BC, Canada, pp.21–30.

García, P., Montalà, O., Pairot, C., Rallo, R. and Skarmeta, A.G. (2002) 'MOVE: component groupware foundations for collaborative virtual environments', *Proceedings of the ACM Fourth International Conference on Collaborative Virtual Environments*, Bonn, Germany, pp.55–62.

Gelernter, D. and Carriero, N. (1992) 'Coordination languages and their significance', *Communications of ACM*, Vol. 35, No. 2, pp.97–107.

Geyer, W., et al. (2001) 'A team collaboration space supporting capture and access of virtual meetings', *Proceedings of the ACM 2001 International SIGGROUP Conference on Supporting Group Work*, Boulder, CO, pp.188–196.

Gräther, W. and Prinz, W. (2001) 'The social web cockpit: support for virtual communities', *Proceedings of the ACM 2001 International SIGGROUP Conference on Supporting Group Work*, Boulder, CO, pp.252–259.

Greenberg, S. and Rounding, M. (2001) 'The notification collage: posting information to public and personal displays', *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Seattle, WA, pp.514–521.

Grudin, J. (1991) 'CSCW: the convergence of two development contexts', *Proceedings of the ACM Conference on Human Factors in Computing Systems*, New Orleans, LA: ACM Press, pp.91–97.

Grudin, J. (1994) 'Computer-supported cooperative work: history and focus', *IEEE Computer*, Vol. 27, No. 5, pp.19–26.

Grudin, J. and Poltrock, S. (1997) 'Computer-supported cooperative work and groupware', *Advances in Computers*, Vol. 45, pp.269–320.

Grundy, J.C., Apperley, M.D., Hosking, J.G. and Mugridge, W.B. (1998) 'A decentralized architecture for software process modeling and enactment', *IEEE Internet Computing*, September/October, pp.53–62.

HardSoftCost (2002) Available at: http://e-meetings.mci.com/meetingsinamerica/uswhitepaper.php3.

Hariri, S. and Mutlu, H. (1995) 'Hierarchical modeling of availability in distributed systems', *IEEE Transactions on Software Engineering*, Vol. 21, No. 1, pp.50–58.

Hawryszkiewycz, I.T. (2001) 'A metamodel for virtual enterprises', *Proceedings of the IEEE Workshop on Information Technology for Virtual Enterprises*, Queensland, Australia, pp.91–97.

Haynes, S.R., Purao, S. and Skattebo, A.L. (2004) 'Situating evaluation in scenarios of use', *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW)*, 6–10 November, Chicago, IL, pp.92–101.

Hearst, M.A. (1999) 'The changing relationship between information technology and society', *IEEE Intelligent Systems*, January/February, pp.8–17.

Hengartner, U. and Steenkiste, P. (2004) 'Implementing access control to people location information', *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies*, 2–4 June, Yorktown Heights, NY, pp.11–20.

Hill, J. and Gutwin, C. (2003) 'Awareness support in a groupware widget toolkit', *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, 9–12 November, Sanibel Island, FL, pp.258–267.

Hill, R.D., Brinck, T., Rohall, S., Patterson, J. and Wilner, W. (1994) 'The rendezvous architecture and language for constructing multiuser applications', *ACM Transactions on Computer–Human Interaction*, Vol. 1, No. 2, pp.81–125.

Hoare, C.A.R. (1998) 'Communicating sequential process', *Communications of the ACM*, August, pp.666–677.

Hodel, T.B., Gall, H. and Dittrich, K.R. (2004) 'Dynamic collaborative business processes within documents', *Proceedings of the 22nd ACM Annual International Conference on Design of Communication: The Engineering of Quality Documentation (SIGDOC)*, 10–13 October, Memphis, TE, pp.97–103.

Horn, D.B., Finholt, T.A., Birnholtz, J.P., Motwani, D. and Jayaraman, S. (2004) 'Six degrees of Jonathan Grudin: a social network analysis of the evolution and impact of CSCW research', *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW)*, 6–10 November, Chicago, IL, pp.582–591.

Hovav, A. and Mandviwalla, M. (1998) 'Social behavior in professional meetings: a video analysis of a panel discussion', *Proceedings of the ACM 1998 Conference on Computer Personnel Research (CPR)*, Boston, MA, pp.159–162.

IDEF0 (1993) 'Announcing the standard for integration definition for function modeling (IDEF0), FIPS PUBS, National Institute of Standards and Technology', Available at: http://www.idef.com/Downloads/pdf/idef0.pdf.

J2EE (1999) Available at: http://java.sun.com/j2ee.

Jamali, N., Thati, P. and Agha, G.A. (1999) 'An actor-based architecture for customizing and controlling agent ensembles', *IEEE Intelligent Systems*, March/April, pp.38–44.

Jeffay, K., Lin, J.K., Menges, J., Smith, F.D. and Smith, J.B. (1992) 'Architecture of the artifact-based collaboration system matrix', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.195–202.

Jiang, J., Yang, G., Wu, Y. and Shi, M. (2002) 'CovaTM: a transaction model for cooperative applications', *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC)*, Madrid, Spain, pp.329–335.

Johnson, P. (1992) 'Supporting exploratory CSCW with the EGRET framework', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.298–305.

Johnson, P., May, J. and Johnson, H. (2003) 'Introduction to multiple and collaborative tasks', *ACM Transactions on Computer–Human Interaction (TOCHI)*, Vol. 10, No. 4, pp.277–280.

Jouppi, N.P. (2002) 'First steps towards mutually-immersive mobile telepresence', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.354–363.

Karjoth, G., Lange, D. and Dshima, M. (1997) 'A security model for aglets', *IEEE Internet Computing*, Vol. 1, No. 4, pp.68–77.

Karnik, N. and Tripathi, A. (1998a) 'Agent server architecture for the Ajanta mobile agent system', *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pp.63–73.

Karnik, N. and Tripathi, A. (1998b) 'Design issues in mobile-agent programming systems', *IEEE Concurrency*, July–September, pp.52–61.

Kazanis, P. and Ginige, A. (2002) 'Asynchronous collaborative business process modeling through a web forum', *Proceedings of Seventh Annual CollECTeR Conference on Electronic Commerce*, Melbourne, VIC, Australia.

Kirda, E., Fenkam, P., Reif, G. and Gall, H. (2002) 'A service architecture for mobile teamwork', *Proceedings of the 14th ACM International Conference on Software Engineering and Knowledge Engineering*, 15–19 July, Ischia, Italy, pp.513–518.

Klinker, G., Linster, M. and Yost, G. (1995) 'Cooperative systems for workgroups', *IEEE Expert*, Vol. 10, No. 3, pp.39–40.

Koskelainen, P., Schulzrinne, H. and Wu, X. (2002) 'A SIP-based conference control framework', *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Miami, FL, pp.53–61.

Kosoresow, A.P. and Kaiser, G.K. (1998) 'Using agents to enable collaborative work', *IEEE Internet Computing*, July/August, pp.85–87.

Kouramajian, V. (1995) 'Consortium: a framework for transactions in collaborative environments', *Proceedings of the ACM 1995 International Conference on Information and Knowledge Management (CIKM)*, pp.260–265.

Kouzes, R., Myers, J.D. and Wulf, W.A. (1996) 'Collaboratories: doing science on the internet', *IEEE Computer*, Vol. 29, No. 8, pp.40–46.

KQML (1993) In E.I.W. Group (Ed). *Specification of the KQML Agent Communication Language*, DARPA Knowledge Sharing Initiative.

Krowne, A. and Bazaz, A. (2004) 'Authority models for collaborative authoring', *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS)*, 05–08 January, Big Island, Hawaii, pp.10018–10024.

Kuutti, K. (1991) 'The concept of activity as a basic unit of analysis for CSCW research', *Proceedings of the Second European Conference on Computer Supported Cooperative Work*, Amsterdam: Kluwer Academic Publisher, pp.249–264.

Labrou, Y. and Finin, T. (1999) 'Agent communication languages: the current landscape', *IEEE Expert*, March/April, pp.45–52.

Lamming, M., Eldridge, M., Flynn, M., Jones, C. and Pendlebury, D. (2000) 'Satchel: providing access to any document, any time, anywhere', *ACM Transactions on Computer–Human Interaction (TOCHI)*, Vol. 7, No. 3, pp.322–352.

Lander, S.E. (1997) 'Issues in multiagent design systems', *IEEE Expert*, pp.18–26.

Laurillau, Y. and Nigay, L. (2002) 'Clover architecture for groupware', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, 16–20 November, New Orleans, LO, pp.236–245.

Lee, B.G. (1998) 'An integrated approach to version management, role-based access control and history maintenance in computer supported collaborative writing', *Department of Computer Science and Engineering*, Auburn University.

Lee, B.G., Chang, K.H. and Narayanan, H. (2001) 'An integrated approach to distributed version management and role-based access control in computer supported collaborative writing', *The Journal of Systems and Software*, Vol. 59, No. 2, pp.119–134.

Lee, J.H., Prakash, A., Jaeger, T. and Wu, G. (1996) 'Supporting multi-user, multi-applet workspaces in CBE', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, November, pp.344–353.

Li, D. and Li, R. (2002) 'Transparent sharing and interoperation of heterogeneous single-user applications', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.246–255.

Li, D. and Muntz, R. (1998) 'COCA: collaborative objects coordination architecture', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.179–186.

Li, D. and Muntz, R.R. (2000) 'A collaboration specification language', *Proceedings of Second Conference on Domain-Specific Languages*, Austin, TX, pp.149–162.

Li, D. and Patrao, J. (2001) 'Demonstrational customization of a shared whiteboard to support user-defined semantic relationships among objects', *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, Boulder, CO, pp.97–106.

Li, F.W.B., Li, L.W.F. and Lau, R.W.H. (2004) 'Supporting continuous consistency in multiplayer online games', *Proceedings of the 12th Annual ACM International Conference on Multimedia*, 10–16 October, New York, NY, pp.388–391.

Li, Q. and Lochovsky, F.H. (1998) 'ADOME: an advanced object modeling environment', *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 2, pp.255–276.

Logrippo, L., Faci, M. and Haj-Hussein, M. (1992) 'An introduction to LOTOS: learning by examples', *Computer Networks and ISDN Systems*, Vol. 23, No. 5, pp.325–342.

Luckham, D.C., Kenney, J., Augustin, L., Vera, J., Bryan, D. and Mann, W. (1995) 'Specification and analysis of system architecture using rapid', *IEEE Transactions on Software Engineering*, Vol. 21, No. 4, pp.336–355.

Malone, T.W. and Crowston, K. (1990) 'What is coordination theory and how can it help design cooperative work systems', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.357–370.

Mark, G. (1997) 'Merging multiple perspectives in groupware use: intra- and intergroup conventions', *Proceedings of the ACM International SIGGROUP Conference on Supporting Group Work: The Integration Challenge*, pp.19–28.

Mark, G., Haake, J.M. and Streitz, N.A. (1996) 'Hypermedia structures and the division of labor in meeting room collaboration', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.170–179.

MCI2002 (2002) Available at: http://e-meetings.mci.com/meetingsinamerica/articles/MIA4_pressrelease.php.

MCI2003 (2003) Available at: http://e-meetings.mci.com/meetingsinamerica/pdf/MIA5.pdf.

McLaughlin, B. and Loukides, M. (2001) *Java and XML*, 2nd edition, O'Reilly Java Tools.

Medeiros, S.P.J., Souza, J.M.d., Strauch, J.C.M. and Pinto, G.R.B. (2001) 'Coordination aspects in a spatial group decision support collaborative system', *Proceedings of the 2001 ACM Symposium on Applied Computing (SAC)*, Las Vegas, NA, pp.182–186.

Millen, D.R. and Patterson, J.F. (2002) 'Stimulating social engagement in a community network', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.306–313.

Mills, K.L. (1999) 'Introduction to the electronic symposium on computer-supported cooperative work', *ACM Computing Surveys (CSUR)*, Vol. 31, No. 2, pp.105–115.

Minsky, N. and Ungureanu, V. (1997) 'Regulated coordination in open distributed systems', *Proceedings of Second International Conference on Coordination Languages and Models*, September, Berlin, Germany.

Neuwirth, C.M., Kaufer, D.S., Chandhok, R. and Morris, J.H. (1994) 'Computer support for distributed collaborative writing: defining parameter of interaction', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.145–152.

Nicollin, X. and Sifakis, J. (1994) 'The algebra of timed processes', *ATP: Theory and Application Information and Computation*, Vol. 114, pp.131–178.

Objectspace (1997) 'ObjectSpace voyager core package technical overview', Available at: http://www.objectspace.com.

Padhye, J. and Kurose, J. (1999) 'Continuous-media courseware server: a study of client interactions', *IEEE Internet Computing*, March/April, pp.65–71.

Paoli, F.D. and Sosio, A. (1996) 'Requirements for a layered software architecture supporting cooperative multi-user interaction', *Proceedings of IEEE International Conference on Software Engineering (ICSE)*, pp.408–417.

Plale, B., Eisenhauer, G., Schwan, K., Heiner, J., Martin, V. and Vetter, J. (1998) 'From interactive applications to distributed laboratories', *IEEE Concurrency*, April–June, pp.78–90.

Powell, A., Piccoli, G. and Ives, B. (2004) 'Virtual teams: a review of current literature and directions for future research', *ACM SIGMIS Database*, Vol. 35, No. 1, pp.6–36.

Prakash, A. and Shim, H.S. (1994) 'DistView: support for building efficient collaborative applications using replicated objects', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.153–164.

Prasad, R.V., Hurn, R., Jamadagni, H.S. and Shankar, H.N. (2003) 'Deployment issues of a VoIP conferencing system in a virtual conferencing environment', *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 1–3 October, Osaka, Japan, pp.150–159.

Preguiça, N., Martins, J.L., Domingos, H. and Duarte, S. (2000) 'Data management support for asynchronous groupware', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, December, Philadelphia, PA, pp.69–78.

Reinert, O., Bucka-Lassen, D., Pedersen, C.A. and Nurnberg, P.J. (1999) 'CAOS: a collaborative and open spatial structure service component with incremental spatial parsing', *Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots*, Darmstadt, Germany, pp.49–50.

Robert, O., Dan, H. and Jeri, E. (1999) *Client/Server Survival Guide*, 3rd edition, Wiley Computer Publishing, John Wiley and Sons. Inc.

Robert III, H.M., Evans, W.J., Honemann, D.H. and Balch, T.J. (2000) *Robert's Rules of Order*, 10th edition, Newly Revised, Perseus Publishing Company.

Rodenstein, R. and Donath, J.S. (2000) 'Talking in circles: designing a spatially – grounded audioconferencing environment', *Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems*, The Netherlands: The Hague, pp.81–88.

Roseman, M. and Greenberg, S. (1996) 'Building real-time groupware with groupkit. groupware toolkit', *ACM Transactions on Computer–Human Interactions*, Vol. 3, No. 1, pp.66–106.

Roussev, V., Dewan, P. and Jain, V. (2000) 'Composable collaboration infrastructures based on programming patterns', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CWCW)*, Philadelphia, PA, pp.117–126.

Rubin, W. and Brain, M. (1999) *Understanding DCOM*, Prentice Hall PTR.

Rumbaugh, J., Jacobson, I. and Booch, G. (1999) *The Unified Modeling Language Reference Manual*, Addison Wesley Longman Inc.

Russel, M., Nitsche-Ruhland, D. and Gunzenhauser, R. (1996) 'An integrating transformation-oriented approach to concurrency control and undo in group editors', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.288–297.

Sabbir, A. and Ravindran, K. (2004) 'User-assisted tools for concurrency control in distributed multimedia collaborations', *Proceedings of the 12th Annual ACM International Conference on Multimedia*, 10–16 October, New York, NY, pp.516–519.

Sato, S. and Murakami, T. (1993) 'Supporting collaboration with loose relationship', *Proceedings of the ACM Conference on Organizational Computing Systems*, pp.52–58.

Scheer, A-W. (2000) *Aris-Business Process Modeling*, 3rd edition, Secaucus, New York, Inc. NJ: Springer-Verlag.

Schulz, S., Rozenblit, J.W., Mrva, M. and Buchenrieder, K. (1998) 'Model-based codesign', *IEEE Computer*, pp.60–67.

Shapiro, D. (1994) 'The limits of ethnography: combining social sciences for CSCW', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Chapel Hill, NC, pp.417–420.

Shen, C., Lesh, N.B., Vernier, F., Forlines, C. and Frost, J. (2002) 'Sharing and building digital group histories', *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work (CSCW)*, New Orleans, LA, pp.324–333.

Shipman, I.F.M. and Mccall, R. (1994) 'Supporting knowledge-base evolution with incremental formalization', *Proceedings of the ACM Conference on Human Factors in Computing Systems: Celebrating Interdependence*, Boston, MA, pp.285–291.

Silberschatz, A. and Galvin, P. (1998) *Operating System Concepts*, 5th edition, John Wiley and Sons Publishing Co.

Singh, M.P. (1998) 'Agent communication languages: rethinking the principles', *IEEE Computer*, pp.40–47.

Srinivasan, F.S. (1999) 'Design patterns in object-oriented frameworks', *IEEE Computer*, pp.24–32.

StarTeam (2004) Available at: http://www.borland.com/starteam.

Sturman, D.C. (1996) 'Modular specification of interaction policies in distributed computing', *Computer Science*, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL.

Subramanian, S. and Malan, G.R. (1999) 'Software architecture for the UARC web-based collaboration', *IEEE Internet Computing*, March-April, pp.46–54.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R. and Wenke, D. (2002) 'OntoEdit: collaborative ontology engineering for the Semantic Web', *Proceedings of the First International Semantic Web Conference 2002 (ISWC), LNCS 2342*, Springer, pp.221–235.

Swarts, J. (2004) 'Cooperative writing: achieving coordination together and apart', *Proceedings of 22nd ACM Annual International Conference on Design of Communication: The Engineering of Quality Documentation*, 10–13 October, Memphis, TE, pp.83–89.

Swing (1999) Available at: http://java.sun.com/products/jfc/index.jsp.

Thomas, A. (1997) 'Enterprise JavaBeans server component model for java', Available at: http://java.sun.com/products/ejb/white_paper.html.

Trevor, J., Rodden, T. and Mariani, J. (1994) 'The use of adapters to support cooperative sharing', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.219–230.

Trevor, J., Rodden, T. and Smith, G. (1998) 'Out of this world: an extensible session architecture for heterogeneous electronic landscapes', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, Seattle, WA, pp.119–128.

Venkatasubramanian, N. and Carolyn, T. (1995) 'Reasoning about meta level activities in open distributed systems', *Proceedings of 14th Annual ACM Symposium on Principles of Distributed Computing*, Ottowa, Ontario, Canada, pp.144–152.

Vin, H.M. and Chen, H.M. (1992) 'System support for computer mediated multimedia collaborations', *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp.203–209.

Wainer, J. and Braga, D.P. (2001) 'Symgroup: applying social agents in a group interaction system', *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, Boulder, CO, pp.224–231.

Watson, R.T., Ho, T.H. and Raman, K.S. (1994) 'Culture: a fourth dimension of group support systems', *Communications of the ACM*, Vol. 37, No. 10, pp.44–55.

White, J.E. (1995) 'Mobile agents', *White Paper*, Available at: http://www.genmagic.com.

Will, U.K. and Hicks, D.L. (2000) 'Requirements for development of hypermedia technology for a digital library supporting scholarly work', *Proceedings of the 2000 ACM Symposium on Applied Computing (SAC)*, Como, Italy, pp.607–609.

Wong, D., Paciorek, N., Walsh, T., DiCelie, J., Young, M. and Peet, B. (1997) 'Concordia: an infrastructure for collaborating mobile agents', *Proceedings of Mobile Agents: First International Workshop MA'97*, April, Lecture Notes in Computer Science 1219, Springer-Verlag, pp.86–97.

Yang, G. (2002) 'A uniform meta-model for modeling integrated cooperation', *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC)*, Madrid, Spain, pp.322–328.

Yang, Y., Sun, C., Zhang, Y. and Jia, X. (2000) 'Real-time cooperative editing on the internet', *IEEE Internet Computing*, Vol. 4, No. 3, pp.18–25.

Yen, C., Li, W.J. and Lin, J.C. (2003) 'A web-based collaborative, computer-aided sequential control design tool', *IEEE Control Systems Magazine*, Vol. 23, No. 2, pp.14–19.

Zhang, J., Chang, C.K. and Chung, J-Y. (2003) 'Mediating electronic meetings', *Proceedings of the IEEE 27th Annual International Computer Software and Applications Conference (COMPSAC)*, 3–6 November, Dallas, TX, pp.216–221.