Formalization of Computer Supported Cooperative Work Applications

Carl K. Chang, Jia Zhang, and Tsang Ming Jiang*
Department of Computer Science (M/C 152)
University of Illinois at Chicago
Chicago, Illinois 60607
chang@uic.edu
* Jiang is currently affiliated with AccessLAN Communications Inc.

Abstract

From its first appearance in 1984, Computer Supported Cooperative Work (CSCW) has attracted a lot of research attention in distributed systems. Over the past decade, the CSCW community has developed a number of experimental systems and commercial products to support collaboration. However, much of the reported work is highly application-specific. There generally lacks a sound infrastructure to support the development of a wide variety of CSCW applications. It is essential to formalize the collaboration concepts, model the cooperative and communicative activities, erect CSCW-oriented architectures, and construct a flexible and extendable development environment for building CSCW applications.

1. Evaluation and Projection of CSCW Technologies

From the first appearance of CSCW [1] until now, 15 years have passed. Researchers have already done a lot of work [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. In the broadest sense, CSCW is the application of computing and communicating technologies to facilitate cooperation and collaboration among people. To summarize, we believe that there are two main directions for further work. First is to build a generic model for the collaboration work, such as a general architecture or a formal language to describe properties and conditions in the system. The second direction is to research how to improve system support for many aspects of cooperative work, such as awareness, flow control, coordination, etc. We summarize the past and current solutions to each issue in a table, as shown in Table I. In the table, we also predict the future trend for each issue. However, after inspecting these topics, we find that these design issues span a very broad range of sciences and technologies (many fields in computer science, social science, and psychology). We will exclude some of the issues from our consideration. These include issues such as application style, perspective, social conduct, human proxy, storage management, and multimedia/hypermedia visualization. These topics are strongly related to some other research domains. Application style is an application issue. Perspective is an HCI issue. Social conduct is mainly a sociology issue. Human proxy falls in Artificial Intelligence (AI) area. . Storage management is one of major issues in distributed computing. Multimedia/hypermedia and visualization are mostly graphics and HCI related topics. Thus, they are beyond the scope of this paper.

We will first discuss the past, current, and future trends for each issue. Control issue includes access control, awareness, session control, and workflow control. We will discuss them separately.

For access control current the solution is based on access matrix to shared artifacts. Developments in this area concentrate on more refined control on each shared artifact. In the past, access control is based on a linear access matrix which shows the users' right to access each shared object. Normally, this specification is quite simple and flat. For each shared object, the access right for every user is the same. And the control granularity is coarse. Currently, the access matrix has been improved to include hierarchical structures. System developers can define different fine-grained access right on each shared object for each user. It would be ideal in the future to allow privileged users to define and modify access rights on demand.

Proceedings of the Eighth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'01)
1071-0485/01 $17.00 © 2001 IEEE

TABLE I Paradigm Shifts in CSCW Research

		Past	Now	Future
APPLICATION STYLE		single-user / asynchronous	multi-user / synchronous	community based / ubiquitous
CONTROL	Concurrency	single-thread	multi-thread	parallel-thread
	Access	linear access matrix	hierarchical access matrix	on-demand based
	Awareness	lack of awareness	forced/customizable	
	Session	rigid/predefined	flexible/dynamic	
	Workflow	linear/centralized	multi-dimensional/ distributed	
ENVIRONMENT		single context	multiple context	ubiquitous
PERSPECTIVE		per task	per collaboration	per policy
COMMUNITY		isolated	cooperative	evolutionary
COORDINATION		tightly coupled with computation	loosely coupled with computation	fine-grained, per-service based dynamic composition of coordination/computation
SOCIAL CONDUCT		community-sanctified etiquette	computer-aided virtual community netiquette	federation of colonies of varying cultures and morals
HUMAN PROXY		static/mechanic	mobile/intelligent	continuous self-corrected true proxy
POLICY		predictable pattern/ hard-coded	unpredictable pattern/ meta-coded	cognitive/ policy patterns
STORAGE SCHEMES		heavy-weight/ light-weight (trade-offs) fat client	applet/plug-in (alternatives) thin client	auto-presence / self purge & self migration archive
MULTIMEDIA/ HYPERMIDIA		text/ graphics	multimedia with A/V	virtual reality
FORMALIZE	Architecture	function/structure based	object / loosely coupled actor-based	context-constrained objects / tightly coupled actor-based
	Language	system-centric	knowledge-centric	language-free
	Interface Language	brute-force glueware	middleware	componentware

For awareness control in distributed computing, in the past, there was a lack of awareness. The reason is obvious, since awareness needs system support, and in the past time, even distributed applications were mostly asynchronous collaborations. Technical problems, such as network bandwidth, are the main obstacle. With the rapid development of Internet technology, real-time collaboration is not just a dream any more. Awareness is especially important in synchronous cooperation. Collaborators need to be aware of others' work at any time to decide their own actions. Thus awareness is now a mandatory requirement. At present, awareness has already been enhanced to the extent that users can customize the awareness percentage they want others to know about their work.

For session control, improvements are made from rigid and predefined control to flexible and dynamic control. In the past, session control was pre-regulated from the developers of the system and could not be changed "after the fact". Now the developers may still institute the session control in advance. However, they provide the tools to let users of the system easily customize the control measures on the fly.

For workflow control, enhancements are made from linear and centralized control to multi-dimensional and distributed control. In the past, a central server will keep track of the workflow of the application, maybe using a table-like information structure. Now the workflow control becomes much more complicated. For example, applications with multiple threads may run different parts in parallel, thus concurrency control of all of these threads becomes necessary. This kind of control may not be done only by the central server. The responsibility is usually distributed to a group of controller. In a present distributed application, there may be many parallel tasks at various layers in the system architecture running at the same time. All of these kinds of control can no longer be supported by one-dimensional linear control. Multi-dimensional hierarchical control is the solution.

Summarizing the four kinds of control in CSCW applications, we predict that the future trend of control will be driven by demands. The new types of control will all be customizable and modifiable dynamically at the run time.

For environment management, consideration has been changed from single context to multiple contexts. CSCW applications can never be isolated. To make CSCW applications feasible, all surrounding environment features should be carefully considered. With the fast realization of concepts of reusability and component engineering, applications should not be constrained to one specific environment any more. The ideal applications should be ubiquitous in as many as possible environmental contexts, which is the central idea of multiple contexts. One way to achieve that is to develop a generic descriptive formalism suitable for instantiation in multiple contexts for usage.

For community management, the shift is quite big. In the past, a community in distributed computing is isolated from other communities. At present, communities should cooperate to achieve a mutual goal. Inter-community and intra-community communications are among central issues for collaboration systems. With the increasing scale of collaboration, more and more communities may emerge. In other words, the communities participate in the collaboration may not be clear at the beginning of the collaboration. More communities may emerge to join in the collaboration during the lifetime of a CSCW system. How to deal with the dynamic community evolution is predicted to be the future trend in CSCW applications.

Coordination for collaboration is one of the central issues in CSCW research. In the past, coordination was tightly coupled with computation. Currently, it has been replaced by coordination that is loosely coupled with computation. This change can ease the development of computational part and coordination part. Replacement of one module will not affect other modules. In the future, we believe coordination in collaboration can be specified at a more finely grained level and it will be capable of being changed dynamically. Furthermore, with the development of component engineering, we believe that in the future, coordination and computation can be composed into one component dynamically. These components will be composed on a per-service basis. In other words, a future component will be complete for one specific service, with the full support of both computation and coordination modules contained inside the service package.

Enforcement of policies in the CSCW applications has been enhanced from past predictable pattern and hard-coded style to current unpredictable pattern and meta-coded style. In the past, developers of the system always designed policies for the system and then hard-coded the policies into the system. After the system was delivered to the customers, policies could not be modified anymore. Right now, this style is already obsolete. Developers will use meta-code to design the pattern for the policies. Users can customize the policies at run time. In the future, we predict policy patterns will be modeled and they can be cognitive.

2. CSCW Formalization

This paper focuses on the formalization of development techniques for CSCW applications to support a large variety of collaborative work. The formalization work can be divided into six categories. First is the formalization of collaboration rules. Second is the formalization of multimedia artifact-based collaboration actions. Third is the formalization of collaboration architecture. Fourth is the formalization of collaboration description language. Fifth is the formalization of development process of collaboration systems. Sixth is the demonstration via rapid prototyping of collaboration systems as the proof of concepts by applying our own development methodology. Due to space limitations, this paper will first present a taxonomy of the frameworks of collaboration technologies and project the future research trends. It then discusses the majoir issues related to developing a novel generic model to support the development of a wide range of collaboration applications.

Formalization of the system consists of framework design, formal language support, and interface language design. We will discuss them separately.

Framework provides the foundation of a system. Client/Server model has already been widely used. Generally speaking, it is function/structure-based architecture. Recently appearing actor-based model is a network-based architecture.

Formal language support is the gate to let formal architecture design take off the ground. It has been changed from its past system-centric to its current knowledge-centric. In the past, languages were used to only define the work running in the systems being modeled. It is limited to the concrete applications. At present, description languages are not only used to describe the processing steps but also to describe the domain knowledge. It has become much more flexible to support platform-independent and application-independent processing. Future trend, we believe, will be language-free. Users may not need to learn too much detailed syntax specification any more. A set of easily understandable icons and images may help users to construct the system modularly.

Interface language design has been changed revolutionarily. In the past, interface language was very primitive. It was constrained to only describe the interface functionality and accessed intern attributes in the development of a system. At present, the development of middleware has already made the interface language much more important, since all components have to interact through published interfaces. Interfaces defined using an interface language have expanded its usage cycle from development process to the whole life-cycle of the software component. Thus, future trend of interface language should be *componentware*.

CSCW research can be characterized by three key challenges: communication, coordination, and collaboration. Summarizing the above research issues, we can find that they all fall into one of these three directions. If we use one axis to express one direction of research issue, we can get a three-dimensional picture to depict the space containing CSCW research topics. Plotting each research issue along the corresponding axis, we get a 3D view of CSCW issues, as shown in Figure 1.

We label three axes as Communication (CM), Coordination (CD), and Collaboration (CL). Research about awareness, concurrency, and multimedia/hypermedia falls along CM axis. Research about access, social conduct, and policy falls along CD axis. Research about human proxy, community, perspective, workflow, session, and storage schemes falls along CL axis.

These research issues are not completely isolated from each other. There exist relationships between them. We say that there are relationships between two research issues, in the sense that one issue counts on the other's support. However, we find that these relationships are not weighted equally. As shown in Figure 6, we use blocked line to show that there is strong dependency between two research issues, while using dashed line to express there are weak relationship between two research issues. To let the results countable and easy to compare, Table II is drawn as a matrix of research issues' interactions in CSCW realm. We do not consider the relationships between two issues along the same axis, since there are no dependencies among them and they are, in fact, different aspects for one axis.

3. CSCW Layer Structure

While we were investigating these research issues and their relationships along three dimensions, we found another interesting phenomenon. There exists a relationship between the three research directions. As a matter of fact, there is a layered hierarchy between them, as shown in the Figure 2. Research on communication provides the basis for the whole CSCW system, and as such it becomes the bottom layer of the structure. As networks becomes more and more pervasive, future CSCW communication system will be available any time, any where on any devices. On top of the communication

Proceedings of the Eighth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'01)
1071-0485/01 $17.00 © 2001 IEEE

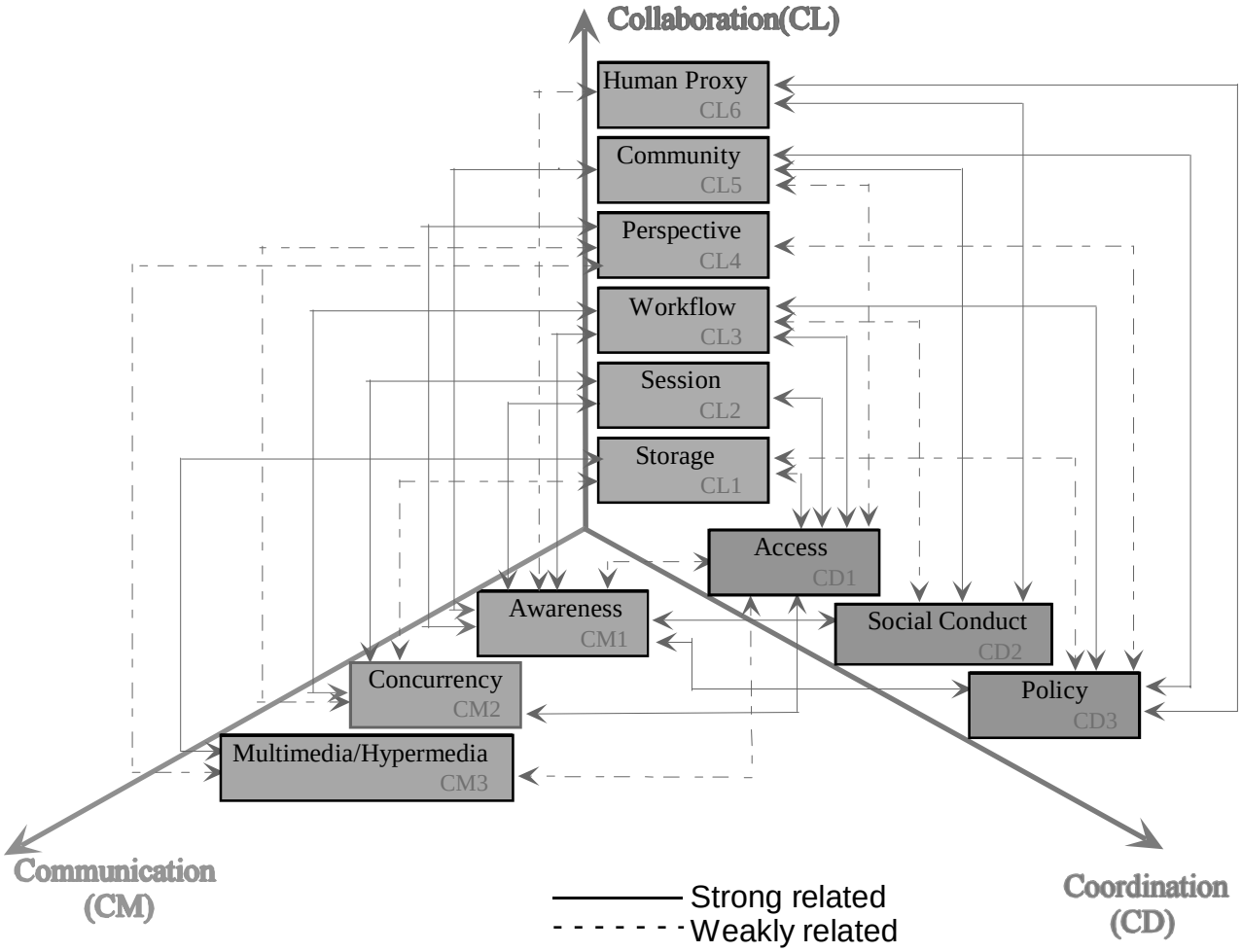

Figure 1. 3-D View of Collaboration Research Issues

layer, research of coordination serves to coordinate the collaborators' work in a CSCW application.

Only with the help of coordination layer, collaborators' work can be integrated together and the whole CSCW application can be effective and efficient. Thus, the coordination becomes the middle layer. Above coordination layer we find the collaboration layer. Collaboration is the final purpose of CSCW. As a matter of fact, CSCW defines an environment for people to collaborate. Thus, collaboration layer becomes the highest layer. From Figure 2, we can also see that the lower layer provides necessary support for its adjacent higher layer so that it serves as the basis for the higher layer.

Remembering the interrelationship we discuss above, this layered structure tells us that, when we consider one design issue in the process of designing CSCW applications, we need to consider together the relative issues in its lower layer. For example, if we like to consider the issue about access in coordination layer, we need to also consider the issues of concurrency, awareness, and multimedia/hypermedia in communication layer.

4. Conclusions

From our survey, we can see that so far, although there are a lot of CSCW systems constructed in the world, much of the reported work is highly application-specific. Each of them does consider some of the issues above, and

provides some solutions to them. However, there is still no current work which comprehensively addresses all or most of the related issues. There generally lacks a sound infrastructure to support the development of a wide variety of CSCW applications. From our analysis above, it is obvious that all issues are unavoidable when facing a CSCW application. A powerful CSCW system should have solutions to all of these issues. Thus, a new infrastructure that supports all CSCW-related issues should be established. Such an infrastructure may be framework-based so that we can conveniently and productively build a new CSCW application. We believe that this framework should be flexible enough so that we can integrate formal rules governing all CSCW-related aspects into the architecture, including perspective, conduct and policies. Meanwhile, the infrastructure should be formalized by providing language support so that we can precisely describe the system, verify the system, and capture the system actions at run rime to keep the system under control. Furthermore, since collaboration work is never isolated from its surrounding environment, resource management and allocation is another important issue we need to consider. Such a profound system framework should not only contain all possible collaboration considerations about every aspect, but it should also address environment and resources requirements.

As mentioned previously, all of the issues we discussed before should be considered when we establish a generic system framework. We summarized these current research issues in CSCW realm in Table I above.. Our research goal is to establish a generic model capable to support a large variety of CSCW applications, and relative methodologies to help users to rapidly develop new CSCW applications. Thus, formalization of CSCW applications is our research objective. It is truly difficult to demonstrate that a model is the best for all kinds of CSCW applications, or most suitable to every category of different applications. Our model can provide solutions to all or at least most of the issues thus raised [12].

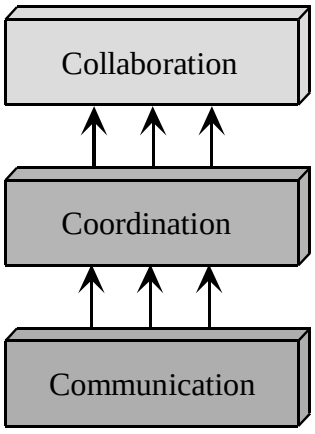

Figure 2. Relationships among Three Dimensions

5. References

1. Grudin, J. and Poltrock, S.: Computer-Supported Cooperative Work and Groupware. Advances in Computers, Vol. 45 pp. 269-320, 1997.
2. Cortes, M. and Mishra, P.: DCWPL: A Programming Language For Describing Collaborative Work. Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work (CSCW), Cambridge MA USA, pp. 21-29, 1996.
3. Astley, M. and Agha, G.A.: Customization and Composition of Distributed Objects: Middleware Abstractions for Policy Management. Proceedings of the ACM SIGSOFT Sixth International Symposium on Foundations of Software Engineering, Orlando, Florida, USA, pp. 1-9, 1998.
4. External Interfaces Working Group: Specification of the KQML Agent Communication Language, Tech. Report, DARPA Knowledge Sharing Initiative, 1993.
5. Li, D. and Muntz, R.: COCA: Collaborative Objects Coordination Architecture. Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW), Seattle Washington, USA, pp. 179-186, 1998.
6. Brusoni, V., Console, L., Terenziani, P., and Pernici, B.: Later: Managing Temporal Information Efficiently. IEEE Expert pp. 56-64, July/August 1997.
7. Fensel, D., Angele, J., et al.: The Knowledge Acquisition and Representation Language, KARL. IEEE Transactions on Knowledge and Data Engineering Vol. 20, No. 4, pp. 527-550, July/August 1998.
8. Arisha, K.A., Ozcan, F., Ross, R., and Subrahmanian V.S.: Impact: A Platform for Collaborating Agents. IEEE Intelligent Systems, pp. 64-72, March/April 1999.
9. Neuwirth, C.M., Kaufer, D.S., Chandhok, R., and Morris, J.H.: Computer Support for Distributed Collaborative Writing: Defining Parameter of Interaction. ACM Proceedings of the Conference on Computer Supported Cooperative Work (CSCW), Chapel Hill, NC, USA, pp. 145-152, 1994.
10. Candan, K.S., Rangan, P.V., and Subrahmanian, V.S.: Collaborative Multimedia Systems: Synthesis of Media Objects. IEEE Transactions on Knowledge and Data Engineering Vol. 10, No. 3. pp. 433-457, May/June 1998.

11. Li, Q. and Lochovsky, F.H.: ADOME: An Advanced Object Modeling Environment. IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 2. pp. 255-276, March/April 1998.
12. Chang, C.K., Quek, F., Lie, C., and Kim, S.: A Research on Collaboration Net. Future Trends of Distributed Computing Systems, Tunis, Tunisia, pp. 228-233, October 1997.

TABLE II Matrix for Issue Interactions

	CM1	CM2	CM3	CD1	CD2	CD3	CL1	CL2	CL3	CL4	CL5	CL6
CM1	X	X	X	○	◎	◎	X	◎	◎	◎	◎	○
CM2	X	X	X	◎	X	X	○	◎	◎	○	X	X
CM3	X	X	X	○	X	X	◎	X	X	○	X	X
CD1	○	◎	○	X	X	X	◎	◎	◎	X	○	X
CD2	◎	X	X	X	X	X	X	X	○	X	◎	◎
CD3	◎	X	X	X	X	X	○	X	◎	○	◎	◎
CL1	X	○	◎	◎	X	○	X	X	X	X	X	X
CL2	◎	◎	X	◎	X	X	X	X	X	X	X	X
CL3	◎	◎	X	◎	○	◎	X	X	X	X	X	X
CL4	◎	○	○	X	X	○	X	X	X	X	X	X
CL5	◎	X	X	○	◎	◎	X	X	X	X	X	X
CL6	○	X	X	X	◎	◎	X	X	X	X	X	X

X: do not care ◎: strongly related ○: weakly related