グループウェア 13−1
（1995. 7. 20）

コーディネーションプロセスモデルに基づく同期型グループウェアの設計

桑名栄二、堀川桂太郎
NTTソフトウェア研究所

協調作業支援システムの設計においては、グループメンバがシステムの中心的役割を果たすので、ユーザおよびグループの振る舞いや協調作業のモデル・理論に基づく設計が重要であると考える。本報告では、同期型グループウェアの設計、構築においてコーディネーションプロセスモデルを適用した事例について報告する。

Coordination Process Model - based Design for Synchronous Group Task Support System

Eiji KUWANA, Keitaro HORIKAWA
NTT Software Laboratories,
NTT

E-mail: {kuwana, horikawa }@slab.ntt.jp
3-9-11 Midori-cho Musashino-shi Tokyo, 180 JAPAN

It is widely recognized that collaborative intellectual work, such as problem solving, decision making and coordination, is a critical part of organization management. Because a user plays a central role in the cooperative system to support teamwork, the way the user interact with the system, and the way the users interact with each other through the collaborative system is a crucial part of the system design process. In other words, it is very important to design according to a model and a theory of group communication and collaboration.

In this paper, we describe a design approach based on the coordination process model and show an actual design and prototype system for supporting real-time group tasks. We also discuss the effectiveness of this approach.

1. Introduction

Computers and computer networks are considered to be key technologies for supporting cooperative work. The relationship between cooperative work and technology has been studied and reviewed in many research areas, including Computer Supported Collaborative Work (CSCW) [e.g., Greif88, Greenberg91], and organizational science [Applegate91] aiming to achieve design systems that support cooperative activity. Because users play the central role in a cooperative system, the way the users interact with the system and the way the users interact with each other through the system is the crucial part of the design processes.

Traditional system designs were based on (1) technology-driven design, (2) rational design, (3) intuitive design, and (4) analogical design [Olson91]. However, these design approaches do not mainly consider user tasks and capabilities. We think that it is important to understand the relationships between users and systems and the relationship between users, and we should apply our understanding and theories to the design of the system.

For the collaboration and coordination model, there are some frameworks and models such as Dix's CSCW Framework [Dix93], McGrath's group task circumplex [McGrath84], and Malone's coordination process model [Malone 91, 94]. We examined the human collaboration model-based design approach to developing a computer supported collaboration system. We applied Malone's coordination process model of groups for system design to evaluate new practical design approaches, and developed a prototype system based on the model for supporting synchronous group tasks. In this paper, we describe a design approach for a synchronous computer supported cooperative system.

2. Design approach and design model for a cooperative work support system

Considering the central role of users in a cooperative system, we think that system design should be based on a collaboration and coordination theory and a model, when developing groupware systems to support cooperative work. We explain why in detail and present a concrete approach in the following.

2.1 Traditional design approaches

Existing systems have mainly been developed intuitively, or with very specific applications in mind. There are various design approaches. The traditional design approaches can be classified as follows [Olson91].

(1) Technology-driven design

System design is based on an existing fundamental technology.

(2) Rational Design

Design is based on prescription and on tradition.

(3) Intuitive Design

Design is based on the designers' intuition.

(4) Analogical Design

Designers develop a system that is similar to some non-electronic system the user is familiar with.

These design approaches have been used in many areas, such as mechanical system design and traditional information system design.

In human and computer interaction (HCI), however, the user plays a central role in the system, so a more recent approach called user-centered design has been proposed and studied. Olson et al. define user-centered system design as (1) regarding participation, observation, and analysis of users at work, (2) utilizing relevant aspects of theory for users, and (3) iteratively designing and testing the system with user involvement [Olson91].

We think that the user-centered design approach is appropriate not only for HCI systems but also for collaboration support systems, because a group and users perform central roles in the collaboration support system. Among the models related to groups (for example, McGrath's model and Malone's Model), we think that Malone's coordination process model and theory is most relevant to the design of collaboration support system and we apply its theory for a real-time groupware system design, because Malone's model is based on a human collaboration and coordination process-oriented viewpoints rather than on collaboration and coordination task types. In later sub sections, we review Malone's coordination process model and show how we applied it.

2.2 Coordination process model

When we design a groupware system to support collaboration, we need a good theoretical framework that explains human communication and collaboration as we mentioned in section 1. Malone [Malone91, 94] proposed the coordination process model (Figure 1) in which workers adjust and work collaboratively according to the following four-layered model.

(1) Perception of common object

When we do cooperative work, we see same physical objects.

(2) Communication

By means of a common language and exchange of messages, we communicate with each other.

(3) Collaboration (e.g., group decision making)

While we communicate with each other, we take collaborative actions to accomplish shared and common goals. For instance, in a design meeting about a software system, we discuss design issues, propose alternatives, and evaluate them.

(4) Coordination

When we do collaborative work, we sometimes come across conflicts such as evaluation and decision conflicts within the group. In that case, we need a coordination process to resolve the conflict.

Collaboration and coordination processes involve processes other than conventional formal information and data communication. We think that this model means that upper-layer processes such as coordination and collaboration need a concrete establishment of the lower processes such as perception of the common object and communication process. In other words, when we provide a collaboration support system, we must carefully consider lower-layer processes support.

Layer	Process	Element
4	Coordination	goal,action,actor, interdependency
3	Collaboration (Decision Making)	goal,action,actor, alternative,evaluation choice
2	Communication	sender,receiver, message,language
1	Perception of Common Objects	actor,object

Figure. 1 Coordination Process Model (Malone)

3. Functionality model based on the coordination process model

In this section, we describe a functionality model of collaboration support systems designed on the basis of Malone's Coordination Process model.

3.1 Functionality model

As a functionality model for a cooperative work support system, we use the Groupware Functionality Model (GFM) [Olson93]. This is a functionality model for synchronous cooperative work support systems such as a real-time group writing task. The goal of GFM is to provide some order to the range of collaborative system functions, and in the course of doing so provide guidance to both the designer and user of such systems.

GFM consists of four function divisions (task activities, interface activities, session activities, and environment activities) for cooperative work support.

When we work on a team, there is a shared work space that contains data objects on which the group is working, e.g., a proposal, a drawing of a widget, or minutes of a meeting. Activities related to these data objects are called task activities, because they are directly related to the group's task.

Interface activities provide group interface functions for group work support; they are not directly related to operations on data. Examples of interface activities are a various group awareness[1] (control joint attention), customization of a user environment, and view control of shared objects.

In the design of collaboration systems, separating data objects from view objects (e.g., interface objects) provides many benefits. First, one can have more than one interface object for the data object. This is important when considering a shared workspace for multiple users. Second, it would permit the interface objects to differ among users, allowing them to customize the interface object not only in its physical characteristics (e.g., window size) but also in the data object's representation (e.g., pie chart versus x-y coordinate graph).

When a group convenes to work through the collaborative system, activities are needed for formulating the session, creating the shared workspaces or invoking other task-oriented activities, coordinating the members' activity. For example, when the shared workspace is formed, the authorization of participants should be checked. The functionality of session management is defined as session activities.

The last functional area is the relationship between the groupware system and its environment. There are two possible interfaces that a collaborative system may have with its environment: (1) an interface between the shared workspace(s) that it manages and group members' personal workspace(s) and (2) an interface between itself and the real "world." Environment activities support these interfaces.

As we stated above, GFM provides the functionality (i.e., "what" functionality) that should be provided for real time collaboration and coordination tasks. However, in developing a collaborative work support system based on GFM, real design tasks are not easy, because we must carefully consider the way ("how") to implement functionality.

We think that some sort of theoretical framework (e.g., the coordination process model) for human collaboration will give *guidelines for "how" to implement functionality.* In the next section, we present a coordination process-based functionality model for the design of real-time collaboration support systems.

3.2 Coordination process-based functionality model.

A collaboration support system has a range of task-oriented functions that will often overlap with the functions found in single user-applications. Of course, in the design of systems there are interactions between the group functions and task functions that often are the major determinants of design decisions.

As we explained in section 3.1, GFM provides a clear explanation of the functionality model even though there is no clear distinction between group functions and task functions. Malone's coordination process model provides some sense of task functions for collaborative work. Thus, we think that an assignment of groupware functionality onto the coordination process model makes a good implementation guide for the collaboration support system. Figure 2 shows the coordination process-based functionality model. In

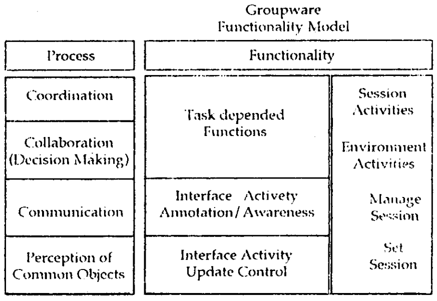


Figure. 2 Functionality Model based on Coordination Process Model

[1] Awareness
A work situation wherein people are cognizant of one another existence, knowing, for instance who the others are and what they are doing, but do not engage in any particular communication or collaborative-work activities.

the following subsections, we describe the assignments of groupware functionality for the development of the collaborative information system.

3.2.1 Supporting perception of common objects

In collaborative work such as meetings, the efficiency of communication among participants depends on the media used. Discussions using documents such as reports, memos, models, blueprints, or video are usually more effective than entirely conversation-based ones.

In the perception of the common object layer of the coordination process model, object operations in a task activity are provided by delivering the same display data to all other participants, known as WYSIWIS (What you see is what I see), and applying the following functions: update control, filter control, format control, window attributes in interface activity of the groupware functionality model.

3.2.2 Communication process support

(1) Support for sharing discussion focusing information

Discussion focus represents who is the speaker, where he (or she) is now on the shared object, and the part of the subject to which he (or she) is giving attention. Therefore a discussion with focus information provided is easier to follow. This corresponds to control joint attention and navigation of interface activity in the functionality model. These functions are usually provided by a tele-pointer or multiple cursors and provide the participants, with information (what we call awareness) of who is in attendance, and where the current discussion focus is, and so on.

(2) Support for sharing annotation messages

Writing annotations (comments, opinions and questions) on some part of a document is a common occurrence in ordinary group work. This function corresponds to annotation of task activity. It has the advantages of allowing supplementary explanations, organization of discussion, and promoting common understanding among participants. Annotation message functions should have flexible user interfaces similar to a ordinary drawing tool.

However, making the user interfaces too flexible and primitive is less advantageous in terms of quick communications, because such a primitive interface requires a number of command steps to write an annotation. Therefore, we analyzed typical user comment patterns. Then we categorized the typical patterns and constructed an annotation model as a tupple <region, link, message> in order to reduce the number of command steps.

Figure 3 shows the annotation model, how annotations are represented, and example of annotation symbols (e.g., regions, links, messages and directions of links). Beforehand participants can select a couple of symbols and attributes for annotation regions, links, and messages. When writing an annotation, a participant specifies the region to annotate and the position for writing a message by pointing with the mouse. Then pre-selected symbols for region and message are displayed. Then the selected link symbols are connected between the two areas. After that the mouse cursor goes to the message position and the status changes to waiting for text input. Participants can thus omit redundant operations and make quick annotations by this automatic annotation function, which treats regions, links, and messages together and provides them as a single annotation object.

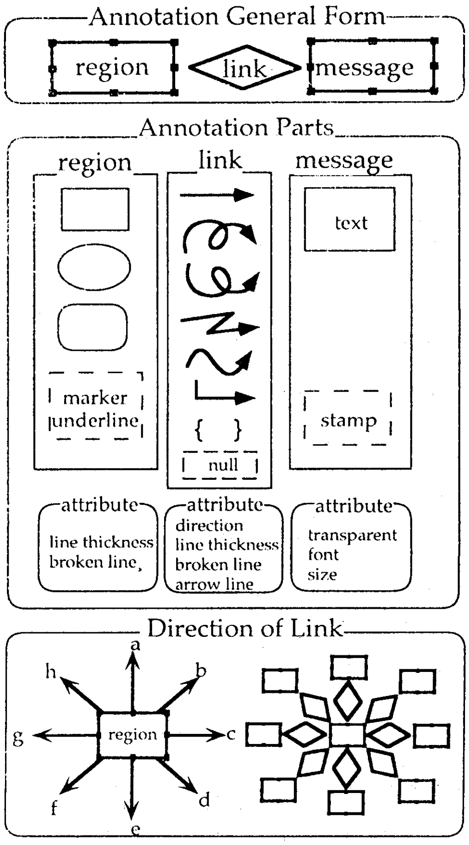


Figure. 3 Annotation object and representation

(3) Support for group decision making and coordination

Functions for group decision making and cooperation support are very different from ones for group tasks. For example, in the case of group design work, such functions as evaluation of design solutions or coordination of conflicting solutions should be supported individually. For example, design rationale and argument structure-based tools will be used for decision making and coordination tasks for software design. However, it is very hard to support all decision making and coordination functions within a single model of functionality, because a decision making process depends on areas of tasks (e.g., software design decision vs. business decision). Therefore these functions are treated as individually provided upon common functions for (1) and (2).

(4) Support for other functions

Session Activity and Environment Activity in the functionality model are management capabilities that create and maintain the states for (1) and (2). Therefore they are also related to all process layers and they are just the same as the ability for managing all process layers in a manner similar to OSI network management. In session management, all functions in session activity, such as support for participants to join or leave the session midway, are realized by managing user information.

4. Implementation of a prototype system

On the basis of the model described in section 3, we developed a shared application system that runs in UNIX and X-window environments. This tool allows users to share various kinds of single-user applications simultaneously. In this section, we describe the implementations based on the functionality model.

4.1 Implementation of the mechanism for sharing discussion subjects

According to the functionality model concerned with the sharing of discussion subjects, the mechanism for sharing discussion subject should be implemented as follows.

The system allows discussion subjects to be shared even if they are represented in different information media. That is, any document can be shared by the mechanism. We introduce functions for sharing any application program by delivering all the application program I/O data. That is to say sharing multimedia (text, graphics and video) applications, as the discussion subject, provide a great effect like "an eye finds more truth than two ears". Therefore, we think that this mechanism improves communication efficiency. This means, as a result, the mechanism lets us expand the range of subject media (e.g., from traditional text to video data as shared objects) corresponding to the perception of the common object layer of Malone's coordination process model. Another effect of the ability to share any application is that each user can use an application he (or she) is familiar with even in groupwork. This helps to provide a the seamless interface between individual work and group work.

4.2 Implementation of communication support

(1) Support for sharing discussion focus

According to the functionality model concerned with the sharing of the discussion focus (i.e., awareness) , the mechanism for sharing discussion focus should be implemented as follows.

Our system allows sharing of user pointing actions related to the discussion subject. Therefore, we introduce functions that realize multiple mouse cursors so that each user can use their own cursor and display it on everyone's windows at any time during the session. The system traces the positions of each participants' cursors and delivers them to other participants and then displays them on each participant's screen. In order to identify who and where the participants are, each cursor's shape and color is unique and the cursor icon has the participant's name and their host machine name. These mouse cursors are displayed on a shared application window as well as on a shared transparent window.

(2) Support for sharing annotation messages

According to the functionality model concerned with the sharing of annotation messages (i.e., annotation) , the mechanism for sharing annotation messages should be implemented as follows.

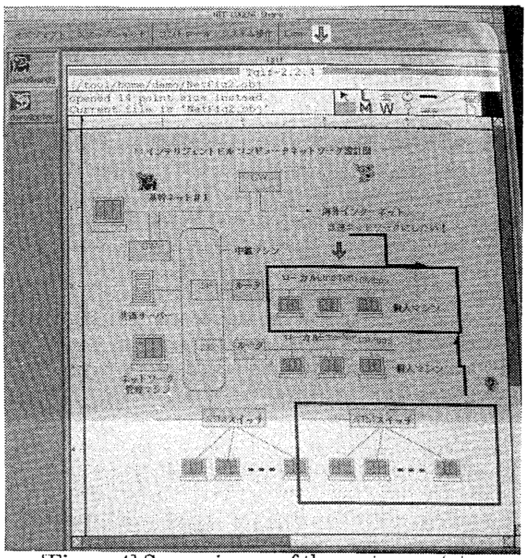
The system should provide a function set that allows all users to write messages relating to the shared discussion subject. Therefore, we introduce functions that provide transparent windows on the shared window to show the discussion subject. Shared application windows are directly overlaid with transparent windows to allow annotation messages corresponding to various parts of the documents to be shared.

Participants can write annotations directly on the application running in real-time rather than by using snapshot operations. We also implement a multiple layer architecture of transparent windows in which one transparent window is prepared for each participant. Therefore, each participant can write annotation messages on their own layer concurrently and independently of others.

4.3 Support for group decision making and cooperation

Because the function for supporting group decision making and coordination depends on group tasks, we think that this function should be provided individually by the domain-dependent shared applications.

Figure 4 shows screen images of the prototype system. In this picture, a drawing tool is shared by a couple of participants. Each participant has his/her own telepointer. While sharing various kinds of tools, participants send and receive communication and collaboration support information such as annotation information.



[Figure 4] Screen image of the protype sytstem

5. Discussion

In this section we discuss the validity and advantages of the system design approach based on the coordination process model, proposed in this paper. We found the following advantages in constructing a collaborative task support system:
(1) Function set arrangement and selection are easier when designing new mechanisms, because the required functions corresponding to each layer in a hierarchical process model are clearly categorized in the functionality model. Comparison with functions of other similar systems is also easier.
(2) Navigation knowledge about system design from existing functionality models is available when we design new systems, because GFM and existing functionality models are regarded as a kind of design guidance information set. Such guidance information is allowed in the GFM in the form of function assignment to some layer of the coordination process model.
(3) We can design the system to have higher overall system performance by selecting alternative functions or by enhancing these functions in lower layers. Strengthening functions in lower layers usually improves the performance of functions in higher layers. Using the hierarchical functionality models may help to select and tune such functions.

6. Conclusion

In this paper we proposed a design approach and describe how to apply it to an actual design. In applying this approach we specified functionality models originally based on Malone's Coordination Process Model and then applied the functionality model to design and implement a system (i.e., transparent windows and application integrated sharing system).

The result was easier functionality assignment and implementation than in conventional approaches. Future work will enhance its approach and apply it to other systems for supporting cooperative group work.

[References]

[Applegate91] L. Applegate, et al., Organizational Computing: Definition and Issues, Journal of Organizational Computing, Vol. 1, No.1, (1991)
[Dix93] A. Dix,"Computer Supported Cooperative Work: A Framework", Design Issues in CSCW, pp.9-26, Springer-Verlag,(1993)
[Greif88] I. Greif (Ed.), Computer-Supported Cooperative Work, Morgan Kaufmann, (1988)
[Greenberg91] S. Greenberg (Ed.), Computer-supported Cooperative Work and Groupware, Academic Press, (1991)
[Ishikawa86] H. Ishikawa, Psychology of Meeting, Chikuma-Book (1986) (In Japanese)
[Malone91] T. Malone. et al., Toward an Interdisciplinary Theory of Coordination," MIT Technical report CCS TR#120 (1991)
[Malone94] T. Malone, et al., The Interdisciplinary Study of Coordination," ACM Computing Surveys, Vol.26, No.1, March (1994)
[McGrath84] J.E. McGrath, "GROUPS: INTERACTION AND PERFORMANCE", Prentice-Hall, (1984)
[Olson91] G. Olson, et al, "User-Centered Design of Collaborative Technology," Journal of Organizational Computing, Vol.1, No.1, (1991)
[Olson93] G. Olson, et al., Designing Software For A Group's Needs : A Functional Analysis of Synchronous Groupware, in User Interface Software (Ed.) L. Bass and P. Dewan, Wiley, (1993)