A Framework of a Support Environment for Cooperative Works over Distributed Computing System based on a Decision Management

Koichiro Ochimizu and Chie Kadowaki
March 24, 1997
IS-RR-97-0013S

School of Information Science
Japan Advanced Institute of Science and Technology, Hokuriku
Asahidai 1–1, Tatsunokuchi
Nomi, Ishikawa, 923-12, JAPAN
ochimizu,kadowaki@jaist.ac.jp

ISSN 0918-7553

A Framework of a Support Environment for Cooperative Works over a Distributed Computing System based on a Decision Management

Koichiro OCHIMIZU and Chie KADOWAKI
Japan Advanced Institute of Science and Technology, Hokuriku
1-1 Asahidai, Tatsunokuchi, Noumi, Ishikawa 923-12, Japan
Phone: +81-761-51-1260
Fax: +81-761-51-1149
e-mail: {ochimizu,kadowaki}@jaist.ac.jp

1 Introduction

In this paper, we will propose the framework for supporting cooperative works over distributed computing systems. Supposing the existence of the ideal distributed computing system that can resolve the problems peculiar to distributed computing systems such as resource sharing, location transparency, consistency management, security, fault-tolerance, and load balancing, there still remain problems to be solved because of the geographical dispersion of participants being engaged in co-operative works. One of them is the management of the state of their co-operative activities. For example, a decision management support is important to make conversations smooth, and managing the state of artifacts/products is also essential to control a software process enactment.

We believe one of the goals of the future SDEs is the enhancement of pleasantness featured by several parameters such as definiteness, independentness, smoothness and portability[1]. **Definiteness** means that we can follow a design method in a concrete and definite manner with an evaluation of design artifacts. **Independentness** or asynchronus mode of works means that we can keep the individual's working environment independent from the others but with necessary communication. **Smoothness** means that every member of the team can communicate with each other smoothly. **Portability** or arbitrary selection of working places means that we can quickly access the necessary information from any places where we will be, or information can follow after us to every place we will visit.

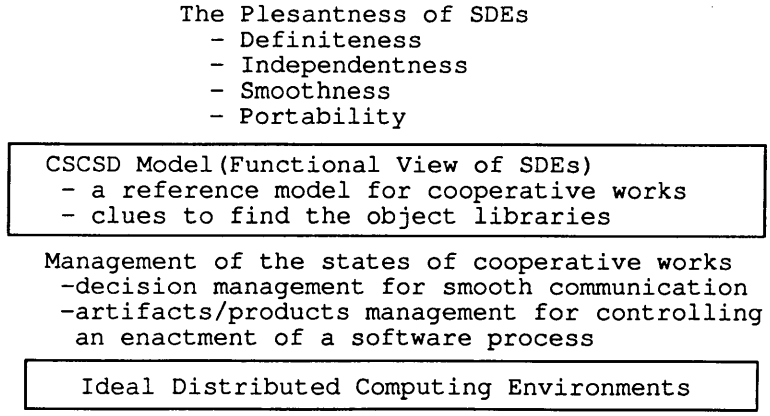

Figure 1: The role of the CSCSD model

In this paper, we will propose a framework(i.e. a functional view of support environments) for the cooperative works, being able to satisfy the goal mentioned above and being able to help us manage

the state of the cooperative works. The framework is called the **CSCSD model**(Computer Supported Cooperative Software Development model). We expect that the CSCSD model will play a role of a reference model of the SDEs for cooperative works over distributed computing systems and also will be able to give us clues to find the object libraries to construct the SDEs.

2 The enhancement of pleasantness of SDEs

2.1 Groupware related issues

- **Merging repository control technologies, especially for decision-management, into the current state of the arts of groupwares is important to make long-term discussions smooth and to help us produce a feeling of togetherness**

Groupwares gave us useful communication tools to conquer time and space distribution of our communication by developing the following technologies: enhancement of realtime synchronous interaction and a feeling of togetherness made by realtime groupwares; proper presentation of information using multi-media; reduction of coordination cost of face-to-face meetings by e-mails. Effectiveness of the asynchronous communication (e.g. e-mail) that guarantee a communication from/to the other without interrupting one's own work is also widely accepted.

However, if they are the only benefits got back from huge investment for networks as a social infrastructure, the benefit is too small. We should explore the new application areas where we can use networks and computers to the full as a tool of intelligence augmentation in the communication.

One of the keys to exploit the new application area exists in dealing with background information of communication explicitly in the foreground. In the software development process, for example, most of the conversations are related to some part of the large amount of documents and/or knowledges of designers/programmers. Although existing groupwares can support well the activities such as noticing, asking, explaining and so on, but as for treatment of the documents, what we can do now is to display one of them on the screen and to manipulate it together. Taking large amount of documents and complex interrelationships among contents of the documents into consideration, capabilities of existing groupwares seem to be insufficient. Groupwares should provide the capabilities that can deal with a large amount of background information explicitly in the foreground, sharing them among participants.

There is a paper that point out this problem clearly. C. Ellis suggests us the research direction of groupwares in his paper[2], by defining and discussing the three user-oriented models of groupwares based on the research results in software modelling. They are; ontological model that is a static description of objects manipulated by users; coordination model that is a structural representation of activities performed by users; user interface model that is an interface representation between a system and users, or among users.

According to the ontological model, we can define necessary static information to be stored in a repository that plays a role of background information during our conversations. No one can know everything. In most of the situations, every participant has partial knowledge on the content of the repository and some of them are shared by the others. The motivation of the communication happens when someone is aware that there exists the gap of the knowledge between/among people. And then conversation would happen to remove the gap. The content of a conversation is understood by the context that each participant has. Structural context is a representation of the dynamic state of a repository. The social context and the organizational context are representation of dynamic mental states of the participants and their social background respectively. The flow of the conversation is controlled by the the context. The procedures, rules and common sense are represented by the coordination model. User Interface Model provides us the related documents, contents of communication, state of the progress of a conversation using visual representation of them.

2.2 Software process modeling related issues

- **Managing the state of artifacts/products is essential to control a software process enactment by formulating tasks and responsibilities for them, and by providing the mechanisms for sharing artifacts/products with network-wide change/failure control supports**

The larger the scale of software is, the more serious the weakness of a human being become. It is difficult to predict and reproduce behaviors of human beings because human being is not a machine and can not make a perfect one at a time. We must take these human factors into account in software process modeling.

In our approach, we put our primary concern in defining the context of one's task instead of writing the procedures they must obey. Admitting the iterations as natural ones, goal of the task(e.g. quality level of products), constraints and regulations not to be violated, inter-relationships of tasks(e.g shared documents) to the other and their priorities are placed around the individual's working environment. Quality level of products, constrains, regulations and relationships are the attributes of task interfaces. One of the important technical issues to be studied is a coordination support mechanism among related people when they are violated.

We should also solve the coordination problem between the individual's activity and the goal of the a team. There is a big gap between the goal of individuals and the goal of the project team. The former admits a self-centered type of work that means one goes on with one's work within one's capability, making oneself consent, to make a good result. The latter requires a self-sacrificial type of work that means one tries to finish one's duty on a limited time. Some one can do both but it is a rare case. We need yet another process model for coordination supports that can bridge between the goal of individuals and the goal of a project team.

2.3 New computing environments related issues

- **Portability of environments or arbitrary selection of working places is also important for improving plesantness**

An infrastructure for the new computing environment such as internetworking, mobile computing environment, ubiquitous information environment are growing now. What kinds of new possibilities will these environment bring us ? In a word, we can quickly access the necessary information from any places where we will be, or information can follow after us to every place we will visit.

2.4 Necessity of the new kind of software tools

- **The SDEs should give us a chance to improve the quality of our work by supporting the acquisition of the unknown resources and the unknown knowledges dynamically**

Technical innovation of information retrieval technologies will occur in near future based on information filtering techniques that support us to find and use only necessary information from huge amount of data scattered over a computer network. Various kinds of information is now being accumulated on world-wide spread nodes of networks due to popularization of WWW. We call these information ubiquitous information. It will be one of important applications that supports exploratory learning or on-demand learning, where a user with limited knowledge on information acquisition can find necessary information from ubiquitous information scattered over a computer network with help of a computer. We call the application(i.e. a tool that support exploral learning) an active channel. A channel means a link for information filtering generated dynamically for each information retrieval request. Term active means that increase in information useful for information retrieval.

3 Supposing the ideal distributed computing system

Distributed systems have several merits, both technical and economic[3];

- Better reflection of inherent geographical dispersion of many applications and their users
- Performance gains achieved by solving complex problems in parallel across multiple computers
- Improved throughput from executing independent jobs concurrently on different processors
- Enhanced resource utilization from sharing information, programs, and processors
- Increased availability from replicating data and program resources

- Extensibility to handle increased workloads by adding processors and storage components incrementally
- Enhanced reliability by virtue of redundant computing resources

Distributed architectures provide advantages such as concurrency, availability, and fault tolerance. However, these benefits are achieved at the price of increased architectural complexity and performance overhead(See the lower part of Fig.2). Research efforts related to these issues can be structured as shown in the upper part of Fig.2.

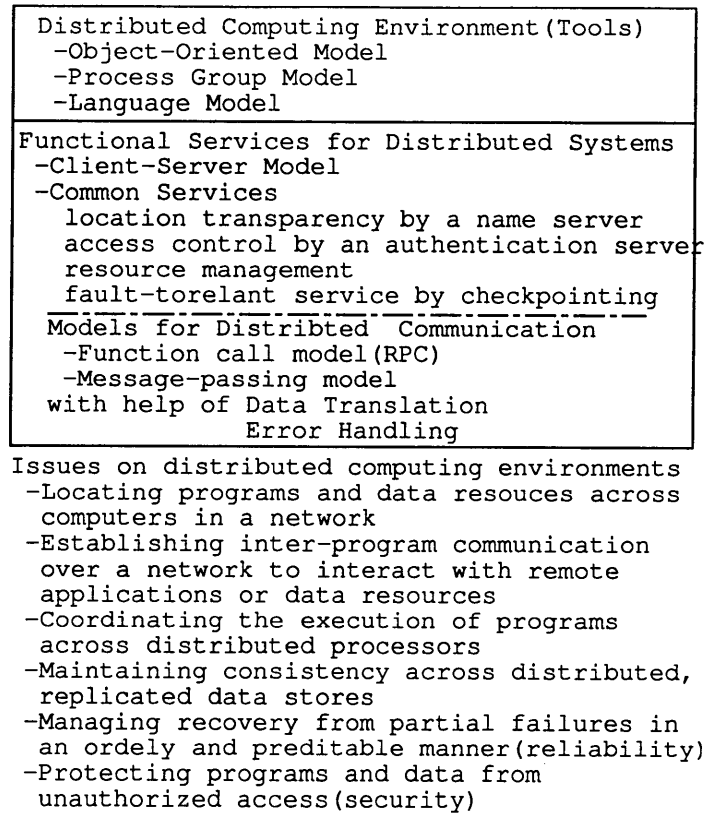

Figure 2: Issues on distributed computing environments and research efforts to them

Supposing the existence of the ideal distributed computing systems that can resolve the problems mentioned above, there would still remain problems to be solved with respect to another level of abstractions such as resource sharing, location transparency, consistency management, security, fault-tolerance, and load balancing among participants being engaged in co-operative works because of the geographical dispersion. We think one of the important issues to be studied is the management of the state of their co-operative works. For example, a decision management support to make conversation smooth, and managing the state of artifacts/products is also essential to control a software process enactment.

- **We need a information repository for cooperative works that can manage the state of artifacts and/or decisions reflecting the state of the outside world of the system exactly.**

4 The CSCSD model for cooperative works

4.1 Integration of a process model and a communication support based on a decision management

We will consider here how to record and manage the decisions, especially for decisions made through conversations, by examining the relationships between products/communication and decisions produced through a software process enactment.

A software development process can be regarded as a decision making process. Everyone manages the decisions made by him/herself based on the range of responsibility of his/her task. For examples, a software designer makes a specification related to design artifacts by him/herself and passes it to a programmer. A programmer designs data structures and control structures of a program referring the specification and then writes a code. The decision (i.e. the specification related to design artifacts) should be shared between the designer and the programmer from the time of delivery. A various kinds of conversation would happen if there is a gap in understanding the content of the delivered thing. Second example is a communication between a project manager and designers/programmers. A project manager should communicate with designers and programmers on the subjects mainly related to the decisions related to time resource management.

Therefore it is important to integrate communication supports into a process model based on a decision management by recording the following information.

- Structural representation of whole decisions
- The state of the progress to the project goal
- Accumulative structure of partial decisions for each subject, and pre-conditions necessary for each partial decision
- Uncertainty and cost of each decision

These requirements suggests that it is necessary for us to develop a repository that can help us manage decisions systematically, from coarse-grained decisions to fine-grained decisions to promote a consensus making and to support change control of decisions. We must also take into account of temporal features of decisions in modeling the repository. In the most of the situations, decisions are temporarily. It means an agreement would become a disagreement and a disagreement would also become an agreement as the time passes and the situation changes.

There are two kinds of decisions in software development activities. One of them is an artifact or a product itself made by a designer or programmer and verified by a test team. The other is a result of choice from design alternatives made through developing an artifact or a product. The former one is formed by structuring the latter ones.

The latter type of decisions, the design rationale, tend to be made by conversations because of low cost of communication. For this reason, the most of researchers take a stand that it is sufficient to record the contents of conversations themselves along the flow of conversations in order to manage design rationales. I think, however, it is redundant because these conversations include a lot of paraphrases and question-answering routines to achieve a good mutual understanding.

It is important for every participant to be able to look at the topic of a current conversation in the proper perspective to the final goal to make communication smooth. A decision tends to be changed so often because human beings can not make a perfect one at a time. So decisions are usually momentary and there is no perfect decision. We need a recording scheme that supports us to change decisions and to analyze the ripple effect of changes.

Decision management means to manage the content mentioned above in addition to the artifacts/products and their dependency relationships. How can we record and manage both types of decisions ? This is the main subject of defining the CSCSD model.

We should refer to, here, the other factors that make communication smooth. They are a paraphrasing process and a feeling of togetherness. One's ability to achieve something usually differs from ability of another person with respect to speed and quality when they are engaged in the same cooperative problem solving activity. It is a usual case that one who achieves a goal first of all explains a point to the others carefully and throughly considering the others' ability. We call it a paraphrasing process. The paraphrasing process also occurs when they are aware of a discrepancy caused by either difference

of understanding or gap of understanding. They try to resolve the discrepancy by paraphrasing their standpoints.

It seems to be useless to record conversations of a paraphrasing process because the contents of the conversation depend on ones' experience, knowledge and physical conditions. Needless to say, it is important to support a paraphrasing process because it helps us form a feeling of togetherness and help us share the state of progress and issues to be discussed next(i.e. what are decided and what are not). We think support of paraphrasing processes is a role groupwares and/or user interface.

4.2 On adopting Coarse-grained dependency-relationships among artifacts as a basis of integration

Modeling and designing activities transform input artifacts into output artifacts/products with referring to materials such as existing products, standardized rules and quality criteria using software tools and/or human beings' thought processes. It means that there are input-output dependency relationships among artifacts/products.

Coarse-grained dependency relationships define input-output relationships among documents such as specifications, design documents and program codes. We can define a software process by adding a temporal order to the coarse-grained dependency relationships among artifacts/products. Fine-grained dependency relationships define use/used or refer-to/referred-to relationships among items included in the documents and give us a basis to understand design rationales. We adopt the coarse-grained dependency-relationships among artifacts/products as a basis for integrating definition of software process models and for recording communications.

We do not think that the coarse-grained dependency is a necessary and sufficient thing as a basis of the integration. It is, however, one of the major basis for the integration, because it provides us constraints in designing the execution order of task and gives us outline of expected conversations.

4.3 On defining a task and a software process based on the coarse-grained dependency-relationships

A subset of artifacts/products is assigned to one of team members according to his/her role and range of responsibility. We call this subset a task. Each task has attributes, for example, rules and forms for describing artifacts/products, constraints on performing the task, interface definitions to the others' tasks, pre/post conditions of the task, quality criteria. As the details of an attribute definition depends on organizations or projects, we consider here the general features common to every task attribute definition to be useful for promoting a cooperation. We think, at least, we need information required by the following functions.

- Constraints not to be violated should be reasonable and should be expressed clearly.
- A system should be able to detect constraint violations.
- A system should be able to give us an instruction in the case of constraint violations.
- A system should be able to support coordinations among related people if a treatment would cause wrong side-effects to the other people,

We define a software process as a lattice of tasks by adding a temporal order to the coarse-grained dependency relationship among artifacts/products. In defining the software process, there is a strong difficulty in balancing time resource assigned to the project and time resource required by the individual depending on one's capability and capacity.

4.4 On recording the content of communication

4.4.1 Recording the progress and the reasoning of deliberations

The flow of conversations represents a transition of participants' concerns. At beginning of the flow, a conversation starts from the the most concerned issue and goes on. In the middle of the conversation, it sometimes occurs to move to the other subject for a time because someone is aware of existing of unsatisfied pre-conditions for the original issue. At the end of the conversation, another issue to be

discussed successively is raised when the original issue is solved. It also the usual case that we can not get to the conclusion during the conversation and suspend it. Talking another issue, we sometime suddenly recognized the solution, then we resume the suspended conversation again. How can we record the decisions made through the flow of conversations mentioned above ? It is easy to record conversation itself along the flow of conversations like a diary. It is, however, the worst style of recording because it can not represent a cause-effect relationship among decisions explicitly.

We often experience the situation during a conversation where we review the conversations so far by arranging what are solved or not and what to be discussed next. We usually arrange the following information at this situation.

- A whole view of decisions made or to be made
- Proper perspective of the current situation in the whole view
- The state of progress for the final goal
- Issues to be discussed next

It is better to record the above four items.

Now we must take it into account that the structure and the content mentioned above may change in progress of the discussion. It also often occurs we can become aware of the imperfection of some decisions later. We should record the following information that can support restructuring of discussion or resteaming of a previous discussion.

- For each subject, an accumulative structure of partial decisions, and pre-conditions required to make each partial decision
- Uncertainty of each decision and cost required for it

Moreover, it is obviously necessary to record

- A correspondence between conversations and software objects

because a subject of conversation is caused by software object.

We introduce here a framework and schematic representations to record the progress and the reasoning of deliberations[4,5]. Our framework is organized into a hierachy consisting of three layers: a deliberation type; a deliberation process; a deliberation space as shown in Fig. 3.

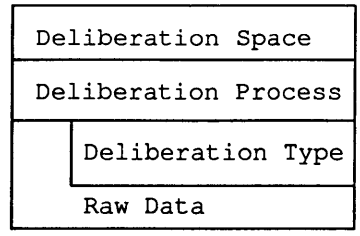

Figure 3: Three layers model

4.4.2 Deliberation space

We represent a cause-effect relationships among decisions to each subject as a deliberation space. The role of a deliberation space is to give us a whole view of decisions and to help us understand both proper perspective of the current conversation to the final goal and the degree of achievement or state of progress to the final goal.

A deliberation space is a graph of deliberation processes connected by several relationships between deliberation processes.

Fig. 4 shows an example of a deliberation space. A rectangular symbol represents a deliberation process. A square symbol also represents a deliberation process. We introduced three kinds of relationship represented by arrows;

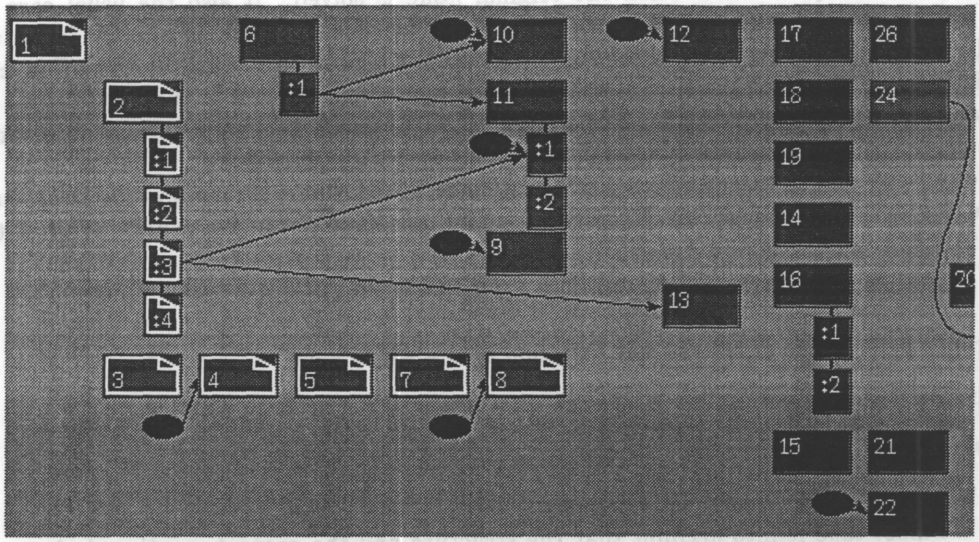

Figure 4: Deliberation Space

succesion a deliberation process follows another deliberation process successively

refinement a deliberation for some subject requires deliberation of several sub subjects. This relationship is represented by an arrow between a rectangular symbol and a square symbol.

co-related Two deliberation process run concurrently sharing some contents

Number assigned to a deliberation process is process ID. Deliberation processes numbered 1 to 8 are at the state " documenting the decisions. Others are in progress. Other state can be distinguished by the colour. By clicking a box, we can examinig the content of utterances grouped by deliberation types. Some of utterances are directly connected to a box. A ellipse symbol is called a "context" that holds the utterances happened outside the groupware used here.

4.4.3 Deliberation process

We manage a series of conversations for subject as a deliberation process. The role of the deliberation process is to manage an accumulative structure of partial decisions and pre-conditions required to make each partial decision by recording conversations made on a subject

There are a lot of subjects to be discussed in a group, and each subject produces a deliberation. Deliberations for subjects are usually interleaved. It is helpful for the paticipants of deliberations to grasp the progress and the reasoning of a deliberation for each subject. We introduce a deliberation process to group the utterances and conversations related to the same subject.

The deliberation process has the state transition as shown in Fig. 5. A deliberation process is created for each subject and is alive during the deliberation. A deliberation is started when several preconditions are satisfied. Examples of preconditions are preparing the necessary documents, attendance of related people, appointed time if any and so on. Several conversation will be held at the state "deliberation". Someone who has the authority declares the end of deliberation and the results(decisions made through deliberation) are documented.

4.4.4 Deliberation type

We manage a series of utterance in a conversation by a deliberation type such as transmitting and adjustment, decision and creation. Most of conversations during a software development process can be represented by the following three types: transmitting some decision or knowledge from the person who know it well to the person who does not know it well; selecting a candidate for the solution from several alternatives by hearing what the other has to say each other; creating some new information by combining the partial knowledges each participant has. The role of a deliberation type is to manage attributes of

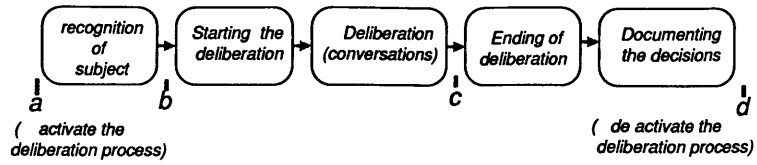

Figure 5: State transition of a deliberation process

a decision such as uncertainty of a decision and cost required for a decision. Each deliberation type has several states and we can judge uncertainty of a decision and/or degree of achievement by seeing at what state the conversation suspended or finished. Time attribute and numbers of utterances can represent a cost of decision.

We categorize conversations into three types based on the work studied by Tanaka[6]: Transmission and Adjustment; Decision; Creation. Each of types explained here is essential one to record the reasoning of conversation, but do not cover the all of the conversation types.

Schematic Representation for Transmission and Adjustment

The purpose of "Transmission and Adjustment" is to reach the state sharing the information or knowledge by performing some question-answering routines. Fig.6 shows a schematic representation for Transmission and Adjustment represented by a state transition diagram.

The conversation initially triggerd by a delivery of something such as an idea, an instruction or an artifact. The reciever examines the content of delivery. If the reciever can understand the content, he/she accepts it and both the transmitter and the reciever become the state "common understanding". If the reciever does not clarify the attitude, there are possibility of disagreement. Then reciever ask something to the transmmitter. At this stage, the transmitter and the reciever become the state "disagreement". Then the effort to fill the gap between transmitter and reciever start, i.e. adjustment, by performing several question-answering routines. The conversation would finish either accepted or disagreement. Former means the success of transmission and adjustment, and the later means the fail of transmission and adjustment.

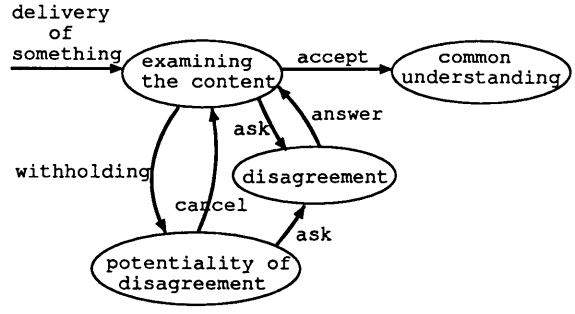

Figure 6: Transmission and Adjustment

Example 1: Delivering an Aritifact

Suppose there exist two persons. Each of them has own role. Fig. 7 shows an example from software development domain. The software designer defines a specification and delivers it to the programmer. In this situation, "Transmission and Adjustment" happens at the initial stage of communication to make a common understanding about the contents of the specification.

It is useful for both people or even the others to record the content of conversation to make clear what is agreed and what is not. This type of information is usually disappeared after delivery, and it makes the situation worse.

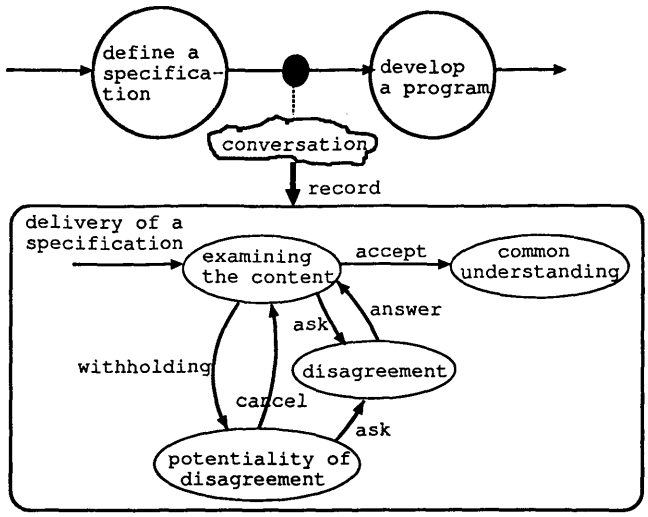

Figure 7: product delivery

Schematic Representation for Decision

The purpose of Decision is to select a proper solution among several candidates for a solution as shown in Fig. 8 This type of conversation starts from need for adjustment among several candidates for a solution.

Initial state is a "existence of multiple candidates". Someone expresses an opinion with reasoning to narrow down the scope of discussion.

By examining the opinion, sometimes it would be rejected, and sometimes it would be accepted. In the former case, another opinion would be expressed. After several discussion, the conversation would be converged and one of the candidates is selected as a soltion.

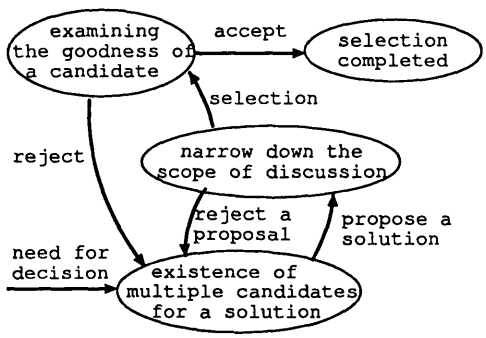

Figure 8: Decision

Example 2: Software Review

The example shown in Fig.9 is also from software development domain. Software review process is a decision making communication process among software designers and reviewers.

Initially, there are two possibilities; acceptance or rejection. Opinions are mainly expressed by revealing all of the defects exhaustively. If the result is rejection, some of those opinions become the reason why the artifact is rejected.

Schematic Representation for Creation

The purpose of creation is to produce something new. Schematic representation for creation is composed by "decision" and "tranmission and adjustment" as shown in Fig. 10.

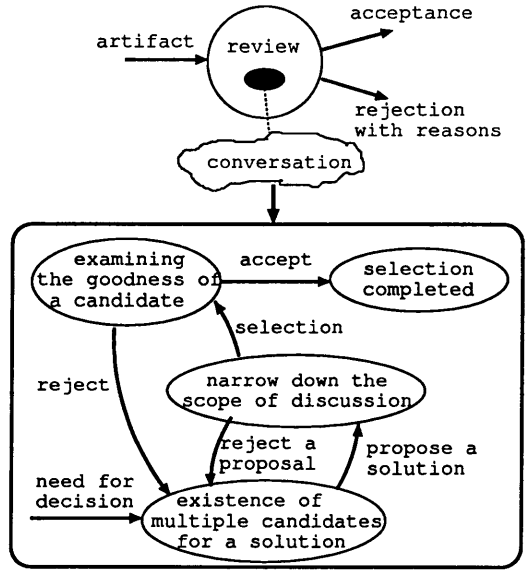

Figure 9: decision

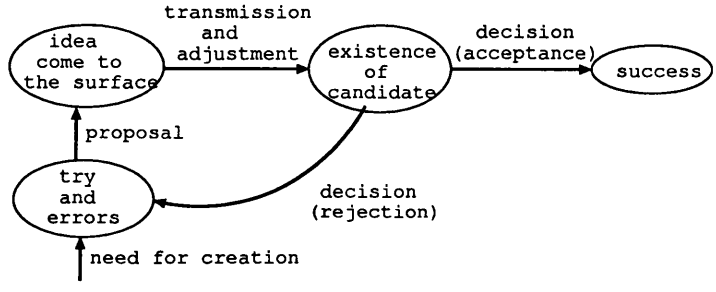

Figure 10: Creation

Combination of Deliberation Types

We can see several combinations of deliberation types as follows.

Decision and Notification

Some of decisions are made by several major staff of the team. After deciding something, the result is notified to the other member of the team to make a global consensus with a lot of transmission and adjustment efforts.

Delivering an Aritifact

In the example shown in Fig. 7, conversation would not end only by transmission and adjustment. If there is severe confrontation between the transmitter and the reciever, the decision type conversation would begin leaving the state of transmission and adjustment "disagreement".

4.4.5 Groupware base

We call a repository with the above scheme **groupware base**. Groupware base is a collection of management pages shown in Fig. 11. A management page is defined for each deliberation process. Item 5 in Fig. 11 keeps information to support resteaming. Item 12 keeps links to instances of deliberation types. Values of Item 14 are completed, prolonged, closed, canceled and not-started.

1	process ID
2	subject
3	goal of a deliberation
4	preconditions for a deliberation
5	resteaming
6	coordinator
7	decision maker
8	participants
9	time resources for a deliberation
10	activation time
11	starting time

12	history of the deliberation(types)
13	deactivation time
14	achivement level
15	the time for documenting the decisions
16	decisions
17	raw data
18	parent process ID
19	child process ID
20	successive process ID
21	co-related process ID

Figure 11: Management page: a recording unit of a groupware base

4.5 On relation between an enactment of a software process and communications occurred

Does a cause-effect relationship among decisions made through conversations have the same structure as an execution order of tasks has ? A structure of a deliberation process does not always match to a structure of an execution order of tasks because some issues happen depending a state of a software development as mentioned later. Therefore we manage a cause-effect relationships among decisions and an execution order of tasks separately.

4.6 the CSCSD model

The central issues for modeling the CSCSD model are as follows:

- Management of tasks that define individuals' responsibility and management of an execution order of tasks(software process) based on the coarse-grained dependency relationship among artifacts/products.
- Management of cause-effect relationships among decisions (groupware base) based on fine-grained dependency relationships among the items included in the documents.

We organize the above functions as a hierarchical structure as shown in Fig. 12.

- At the lowest layer we put the CSCSD server that can help us use various kinds of distributed computing services in an integrated manner(e.g. enabling us a network transparent data access with resolving heterogeneity of data).

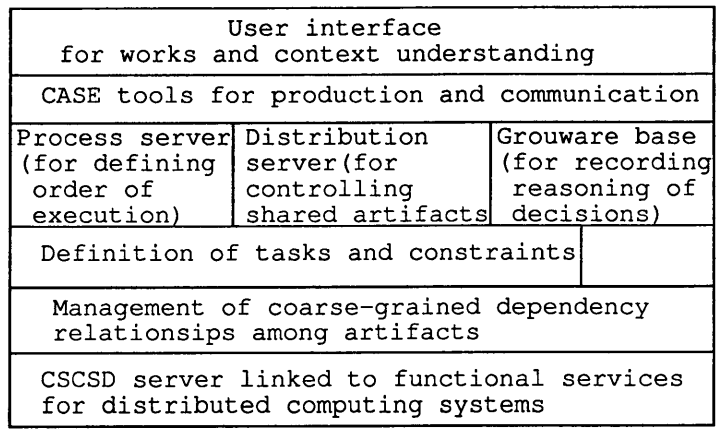

Figure 12: the CSCSD model

- At the first layer from the bottom, we manage the dependency relationships among artifacts/products. Functions of this layer support to define guidelines of works and constraints related to tasks for constructing the distribution server in the upper layer and also give a groupwares the basis for recording the contents of communication by keeping links between contents of conversations and artifacts or dependency relationships of artifacts.
- At the second layer from the bottom, tasks are defined according to one's responsibility with constraints.
- At the third layer from the bottom, we put a process server and a distribution server on them, and a groupware base. The process server defines an execution order of tasks, and resources available. The distribution server defines a group of related artifacts according to one's responsibility(we call it a workspace) and manage shared data between tasks(i.e. an intersection of workspaces)[7,8]. The groupware base keeps a cause-effect relationship among decision made through conversations with respect to tasks and input/output dependency-relationships among artifacts/products[5].
- At the fourth layer from the bottom, we put CASE tools and groupwares to support both producing artifacts and making conversations like consensus making or delegation.
- At the top layer), we put a UIMS to display information related to various kinds of servers and tools. Context information proposed by Ellis will be displayed if necessary.

5 Current state of the progress of JIZAI prototyping

In this section, we will introduce our prototyping efforts. The name of the prototype is JIZAI[7] [1] , which consists of two subsystems named GUNBU [2] and SIORI [3]. The former is a prototype of a distribution server and the latter is a prototype of groupware base.

5.1 A Distribution Server to guarantee Independentness

We are studying management of shared information as one of the necessary conditions to achieve the goal 'independentness'. We have already succeeded in developing a prototype named GUNBU. GUNBU is constructed by two types of objects named a workspace-manager and an autonomous-mediator which can support us to manage shared data in distributed software development[8,9]. The major features of them are as follows;

[1] JIZAI is a Japanese word which means we can do everything freely
[2] GUNBU is a Japanese word which means group dancing
[3] SIORI is a Japanese word which means a bookmark

1. **A workspace-manager object** and an artifact object which can support us(i.e. software engineers) to manage their range of responsibility and control of data sharing.
2. **An autonomous-mediator object** can support negotiation among software engineers occurred through the modification of shared data.
3. Each object has a meta-object that supports us to select an available proper action dynamically according to the situation.

Using these objects, a software engineer will proceed his work, only using both the directly related knowledge to his own responsibility and the direct relationships to the others who share data with him. Our environment will support us to change policies related to data sharing in a cooperative and flexible manner.

Another type of object, named **coordinator** is also being studied as a collaborative work with professor Carlo Ghezzi at Politecnico di Milano. One of the stable ways to control an enactment of a cooperative software process is to observe the state of arifacts/products objects. The coordinator object monitor the states of

- entities that change their states concurrently along a time axis with interacting each other under some constraints on their behavior

Monitoring the state, the coordinator object tries to

- maintain the consistency among the states of several artifact objects(external constraints) and among the states of an artifact object(internal constraints).
- find the culprit of abnormal state by using inference rules that use dependency relationships, constraints, and an activities record of entities.

The coordinator object uses a database that consists of

- a structure used for defining the constraints
- a history record of state transitions of entities and events happened to them

In our modeling,

1. Entities are design artifact/product objects. Each object is formulated as an autonomous object containing a state transition diagram that represents a state of progress of each artifact. Time-related constraints are assigned to each state-transition of STDs to express permitted/ inhibited transitions and exception handling(internal constraints).
2. A structure is an input/output dependency graph among artifact/product objects that is used for defining external constraints.
3. An activities record of artifact objects contains events/states sequences actually happened to them with a version control mechanism.

We now try to define constraints among STDs which can deal with the following semantics.

1. group roll-back: it eliminates all of the wrong side-effects caused by the culprit when it is detected.
2. deviation[10]: some object proceeds its transitions without waiting the occurrence of actual events postulating the happening of proper events. After happening the real event, if some contradictions are observed, it makes roll-back.

5.2 A Groupware Base to provide Smoothness

We have already designed and implemented a prototype system SIORI[11], for electronic meetings using mailing lists, based on the groupwarebase model. Its purpose is to facilitate the coordination among participants and to reduce the possibility of occurrence of redundant conversations caused from an insufficient management of decisions.

Though E-mail and NetNews are useful tools for communication over a network, it still has room to be improved especially for classifying and grasping the communication contents. By adding groupware base to a mailling list tool, we can expect following support functions that enable us to grasp the progress and the reasoning from the vast mails exchanged through a mailling list.

- to understand a deliberation history and decisions made in the past
- to clarify a deliberation condition and a point of an argument in the present
- to coordinate and plan the future discussion to be made

The system consists of "SIORI" server and "SIORI" clients. The "SIORI" server manages a groupware base. The "SIORI" client provides us the functions to refer the groupware base (Fig. 13).

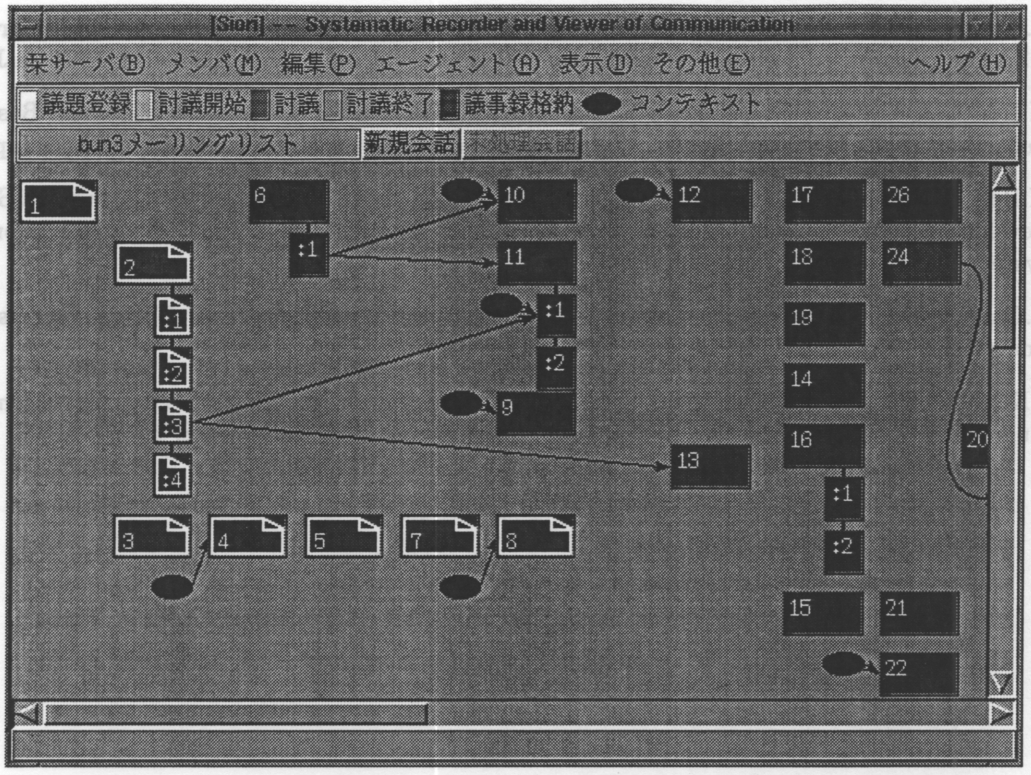

Figure 13: "Shiori" clients

6 Summary

1. JIZAI can support the cooperative works overs distributed computing environments by managing the state of the cooperative works, especially for a decision management
2. CSCSD model will play a role of a reference model for cooperative works over distributed computing systems and also will be able to give us clues to find the object libraries to construct the SDEs.

7 Acknowledgements

I appreciate professor Carlo Ghezzi and Mr. Gianpaolo Cugola at Politecnico di Milano for their useful discussions and comments on the coordinator object.

References

1. K.Ochimizu:"Constructing a Support Environment for Cooperative Works over a Computer Network by Integrating Software Process Enactment Support and Communication Support", Jaist Research Report, IS-RR-97-0012S, ISSN 0918-7553, March 1997.
2. Clarence Ellis, Jacques Wainer: "A Conceptual Model of Groupware", Proc. of CSCW 94, pp.79-88, 1994.

3. R.M.Adler and R.C.Paslay,"Design of Distributed Systems", Encyclopedia of Software Engineering, John Wiley and Sons.

4. C.Kadowaki, K.Ochimizu: "Recording the Progress and the Reasoning of Deliberations".

5. C.Kadowaki, K.Ochimizu:"Systematic Support of Communication in Enacting a Software Process",Jaist Research Report, IS-97-0009S, ISSN 0918-7553, March 1997.

6. H.Tanaka, K.Araki, Y.Masuda:"Relationship between Interpersonal Communication and Effects of Communication Media", IPSJ SIG on GW 93-GW-4, pp.53-60, 1993(in Japanese).

7. K.Ochimizu, C.Kadowaki, K.Fujieda, M.Hori:"Design of A Software Development Environment JIZAI for Distributed Software Development", Technical Report of IEICE SS94-18, 1994 (in Japanese).

8. M.Hori, K.Ochimizu:"Shared Data Management Mechanism for Software Development Based on a Reflective Object-Oriented Model", Computer Software,Vol.13, NO.1, pp.37-54, 1996 (in Japanese).

9. M.Hori, Y.Shinoda, K,Ochimizu: "Shared Data Management Mechanism for Distributed Software Development Based on a Reflective Object-Oriented Model", LNCS 1080, Advanced Information Systems Engineering, pp.362-382, 1996.

10. G.Cugola, E.Di Nitto Frank, C.Ghezzi: "How to deal With Deviations During Process Model Enactment", Proc. of ICSE 17, pp.265-273, April 1995.

11. S.Konno,C.Kadowaki,K.Ochimizu:"Supporting Situation understanding and Documentation of Communication in Electric Meeting by Groupwarebase, Siori", IPSJ SIG on SE, 107-12, 1996.