Flexible support for group interactions in collaborative design

Yongwu Miao and Jörg M. Haake
GMD - German National Research Center for Information Technology
IPSI - Integrated Publication and Information Systems Institute
Dolivostrasse 15, D-64293 Darmstadt, Germany
Email: {miao, haake}@darmstadt.gmd.de

Abstract
Collaborative design systems support cross-functional teams in performing design work collaboratively. However, the sole use of social protocols to coordinate team members’ activities may lead to potentially unexpected interactions and unpredictable conflicts when manipulating shared design artifacts. Therefore, there is a need to provide a computational mechanism to support group interaction in the shared workspace. This paper describes the functionality of SCOPE, a system that provides flexible support for group interaction in collaborative design. Our approach is to provide a mechanism to control the interaction of teams and to dynamically bind group interaction semantics to the properties of a collaborative design software system. Using this approach, group interaction can be controlled in a fine-grained and systematic manner and team members can adapt the system to changing circumstances during collaborative design.

Keywords
Collaborative design, Collaboration protocols, SCOPE, Group interaction, CSCW

1 Introduction
Concurrent Engineering (CE) is a way to speed up product development by integrating downstream concerns as early as possible in the design process and to promote cooperative work of all disciplines and domain experts within cross-functional teams. CE emphasizes both a simultaneous execution of shared tasks and the control of cooperative decision making by cross-functional teams. Therefore, it is important to understand the need for interaction and communication within and between cross-functional teams of individuals who may be scattered over a wide geographic range. In this paper we focus on provision of flexible support for group interaction within a cross-functional team.

In order to support collaborative design in cross-functional teams, most of the existing collaborative design support systems provide desk top audio and video connections and a shared workspace which contains all the shared artifacts used by the team. The shared artifacts are managed and manipulated by all members of the team. Group communication and interaction is performed by using audio and video facilities and by manipulating shared artifacts. However, such systems provide less support for users to organize, coordinate, and control collaborative activities in order to achieve a goal in an organizational context. Team members have to use social protocols to coordinate their activities. Although coordination of interaction is most flexible with vocal agreements, prevention of violations is impossible. That is, potentially unexpected interactions and unpredictable conflicts may occur during manipulating shared artifacts. Therefore, there is a need to provide a computational mechanism to support group interactions in the shared workspace. Such a computational mechanism should be conceived as an abstract device incorporated in domain-specific software applications [7].

Presently, most mechanisms to control the interaction within a cross-functional team are provided either at the data level (e.g., access control) or at the process level (e.g., workflow). However, in some situations, a mechanism provided at only one level is not sufficient to control group interaction [3]. Based on such mechanisms, a coordination policy can be established as a computational definition of group interaction, which is incorporated in the design support software. It will govern actual group interactions in collaborative design processes so as to avoid conflict situations. Coordination policies can be selected and implemented in the program development phase. Thus, users have to follow the built-in coordination policy to interact with each other. However, design is a creative task and collaborative design processes are ill-structured and often long-lived. Such processes vary from project to project and even within the same project at different stages [2]. Therefore, a more dynamic technique is needed for collaborative design systems to adapt to changing circumstances during collaborative design.

This paper describes the functionality of a system which provides flexible support for group interaction in collaborative design. Our approach is to provide a mechanism to control the interaction in teams by combining the data level control mechanisms and the process level control mechanisms, and to bind group interaction semantics to the properties of a collaborative design software system dynamically. The facilities described in this paper have been implemented in the SCOPE (“Session-based
COllaborative Process-centered Environment”) system [5]. SCOPE is an environment that supports specification, modification, monitoring, and execution of session-based collaboration processes.

The remainder of this paper is organized as follows: The next section analyses the problem of group interaction, presents a framework for group interaction, and discusses the issue of flexibility. Section 3 deals with various aspects of the specification of group interaction in collaborative design. In section 4, our approach to support actual group interactions is introduced. Then we compare our approach to related work. Finally, we present our conclusions and plans for future work.

2 Analysis of Group Interaction

McGrath [4] provides a conceptual framework for the study of groups. He describes the group interaction process as the central feature of this model. The major classes of inputs into the group interaction process are properties of the group members, properties of the group structure, properties of the task/situation, and properties of the surrounding environment. These four sets of properties set the conditions under which group interaction takes place. And the effects of these properties, individually and in combination, are forces that shape the group interaction.

In this model, the behavior setting represents a pattern between the group and the task/situation. Both properties of the members and properties of the environment can be used to specify behavior setting directly, as well as indirectly through group and task.

2.1 Framework

The model provided in [4] is very general and abstract. In order to provide support for collaborative design processes, we have to select factors and properties that are thought to be germane to collaborative design. In this section, we propose a framework, which looks into related factors that characterize group interaction processes.

Member. Properties of team members (age, sex, specialty, discipline, work experience, interest, etc) may affect group interaction.

Team. Coordination of interaction depends on the team size and the patterns of relations among members. Normally, the larger the team, the more formal interaction will be adopted. A team may be formed in a way that the team members come from one or more than one (sub) organization having different roles. Thus, role structure and organizational structure should be taken into account.

Design model. The design model is regarded as the most important constraint in collaborative design. A design model is a representational framework that represents the evolving design. The design model forms a basis for knowledge sharing among team members.

Task. Group interaction involves the team carrying out a design task. A task may be decomposable. The decomposition may be hierarchical. A task involves informally assumed goals as well as assigned activities. In case of collaborative design, the activities include design activities and coordination activities. A design activity consists of a potential set of operations that are applied to shared design artifacts (such as create, select, compare, accept, reject, suspend, patch, refine, and so on). These operations may be used to change the content of design artifacts, to change the state of artifacts, or just to navigate among shared artifacts. A coordination activity consists of a potential set of actions to maintain collaboration. The result of an action will change the state of collaboration.

2.2 Flexibility

Flexibility is an overall criterion rather than a property of a certain view of a collaborative design system. Flexibility covers the ability to switch dynamically between different states or modes of a collaborative design. All criteria presented within the framework are now evaluated with respect to flexibility.

A collaborative design system should be flexible to react to interactions with different users in the same way or in different ways. Sometimes, i.e., in a brainstorming session, every team member has the same view of shared artifacts and has the same right to input ideas. However, sometimes it is desirable that system reactions are different for users depending on individual’s properties in collaborative design processes.

Flexibility with respect to a group addresses the dynamics of group size and structure. A flexible collaborative design system should be open to any new participant. The group size should be a dynamic quantity to allow users to join or leave a session when desired. The group size may influence the behavior of the group members. Team members who come from different departments or have different roles may have different influence on each other. A collaborative design system should allow end-users to represent organizational structures and to change them dynamically.

Flexibility with respect to the representation of design product information ranges from supporting unstructured representations to well-structured representations. Structured representations allow for more sophisticated computer-based support for detecting and resolving conflicts between the work of collaborating designers whereas unstructured representations must rely exclusively on human interpretation. Semi-structured representations provide a limited amount of structure usually in the form of information chunks and relationships
between them. However, the contents of these information chunks are described using natural language or other not-interpreted media such as image, drawing, graphics, audio, etc.

Flexibility with respect to the task yields the possibility to support a range from unstructured tasks to well-structured tasks. That is, on the one extreme, design tasks may be carried out in routines which can be ‘programmed’. On the other extreme, design tasks are fulfilled by unpredictable and spontaneous operations. The degree of structuredness of tasks determines the degree of formality of group interactions. Formal interaction handles information that is easy for the computer to analyze or process further. On the other hand, an informal interaction deals with design ideas that are extremely hard to formalize and to automatically process. Especially in engineering processes, the objects of interest are quite informal during the early phases and migrate to more and more formal objects along with the process.

2.3 Group Behavior

Team members may act individually or jointly, synchronously or asynchronously, and routinely or spontaneously. The process of defining group behavior is to bind group interaction semantics with properties of the collaborative design system. Group behavior can be described in terms of process definitions. These definitions specify who can perform what operations and actions in each phase, in what collaborative mode, and in what order. Such process definitions are called collaboration protocols in this paper.

Team design is viewed as a social construction of a technical reality focused on creating and maintaining a shared understanding of the design process and product among the team members [11]. They have to establish their own processes to reach consensus on terminology, constraints, trade-off, and so on. They have to adapt their collaborative policies to fit some changes. Thus, systems should support team members in accessing and manipulating the mechanisms of interaction in such a way that they are able to handle contingencies and adapt the collaboration protocols to changing requirements in their environment.

With dynamic process binding, team members can define or change collaboration protocols at run time. They can change the stipulations about which of the operations and actions can be activated, in what order, and by what roles. Group interaction processes can also be structured to allow some operations and actions to be initiated concurrently, or to be synchronized with the completion of earlier actions.

3 Approach

In this section we present our approach to support group interactions. This approach can be characterized by (1) providing a shared hypermedia workspace, (2) using hypermedia to represent shared design artifacts in a flexible way (i.e. as a design model), (3) using hypertext structures to specify collaboration protocols (i.e. executable descriptions of collaboration behavior), and (4) facilitating the execution of collaboration protocols through the cooperative tools provided to the team members.

This approach has been tested in the SCOPE system. SCOPE is implemented based on the COAST system [8] which has well-developed facilities to maintain a shared workspace, such as fully replicated concurrency control. By combining the data level control mechanisms, such as access control and concurrency control, with process level control mechanisms, group interaction can be controlled in a fine-grained and systematic manner.

Next, we first discuss how to support the specification of group interaction. Then support for group interactions at run time is presented.

3.1 Specification of Group Interactions

As mentioned above, in this paper we focus on provision of flexible support for group interaction in a session. The concept of a session is defined in SCOPE in the following way: In a period of time team members perform activities in a shared information space. Here, a session is created for a cross-functional team to solve a design task collaboratively. In a session, the common information space is potentially accessible to all team members involved in this session. The common information space contains all artifacts used and produced in the session. Team members can perform operations synchronously or asynchronously to construct shared artifacts together. They can also perform actions to change the state of collaboration. It is important to notice that a cross-functional team consists of experts in a variety of disciplines and domains. They may have different privileges and responsibilities to deal with artifacts and to coordinate their efforts to get their work done.

In the following, we discuss how group interaction is specified in SCOPE at definition time. Group interaction in a session is specified in SCOPE by using three tools: organizational structure definition tool, activity space definition tool and collaboration protocol definition tool. These tools are designed with the framework of group interaction in mind, which was presented in the last section.

3.1.1 Definition of Members and Teams

Members and teams are defined by using the organizational structure definition tool. With this tool, a member, an organizational-role or an organizational-unit is represented as a node with a type and a unique name. Organizational-unit refers to divisions within organizations. Organizational-role refers to one person or a group of people in an
organizational context, such as department manager, senior engineer, member of a given project, or consultant. Each type of node (member, organizational-role and organizational-unit) has a set of attributes to describe properties. For example, a member node has attributes such as age, sex, and specialty, discipline, work experience, etc. The "is-a-member-of" link is represented as a direct arrow from a member to an organizational-role or an organizational-unit. Organizational-roles and organizational-units are structured. That is, there are "is-a-sub-role" and "is-a-sub-unit" direct links among organizational-role nodes and among organizational-unit nodes respectively. A description of an organizational structure is represented as a Directed Acyclic Graph (DAG). Each node of a DAG will be an entry for access control. A team can be regarded as an organizational-role or an organizational-unit. The members of a team are recursively determined by the organizational structure.

3.1.2 Definition of the Design Model
A design model is defined as an activity space in SCOPE. An activity space is defined in terms of task-specific typed hypertext objects (nodes and links) together with operations on them to support problem solving in a certain domain [10]. Using a labeled graph representation, it can be regarded as a semantic network that has structural, relational, and computational semantics [12]. Design objects are organized into these node and link types according to their structure, function, and behavior properties. An activity space instance provides support for creating information spaces consisting of instances of allowed object types, limiting its organization to allowed structures, and potentially offering task-specific operations. The allowed object types in an activity space may be basic object types such as String, or may be composites (i.e. other activity spaces). That is, activity spaces are hierarchically structured and different levels can represent different levels of product abstraction. Defined activity spaces are stored in a design model base. This base can be used to define other activity spaces.

As mentioned above, designers may need to define their own design model at run time. The design model is usually transformed gradually from informally structured information space into more formal structures. COWFISH [12] is a system, in which a flexible hypermedia model has been developed for representing activity space semantics. A flexible hypermedia system is a hypermedia system which supports the co-existence and transformation of information structures in different degrees of formality, i.e., from unstructured representations to well-structured representations [1]. In COWFISH, the elements of the hypermedia model are nodes, links, node content pages, and other media objects such as text, drawing, graphics, image, etc. An activity space is represented as a nested node structure. Each node has a content page. The type of an activity space is determined by its root page. The sub-activity spaces of an activity space are defined by the page types of the nodes at each level of the nested node structure. The system provides an example-based definition tool for end-users to create task-specific activity spaces. Thus, a designed product model can be abstracted and then serve as a prototype and as a modeling language. Another tool in this system is the flexible space that allows end-users to develop a pattern gradually using unconstrained hypermedia in order to support emergent structures. It is a helpful tool for a design team, because their shared understanding is built up over the life of a project. Normally, the information and knowledge about a design may initially consist of a web of loosely structured objects. Over time, concepts are gradually transformed into design models, becoming more formal and detailed. Information is abstracted and reorganized concurrently so that team members can share it. The tools of COWFISH have been integrated into SCOPE to support this process.

3.1.3 Definition of collaboration protocols
Using the collaboration protocol definition tool a collaboration protocol is specified. A collaboration protocol is described as a state-transition diagram where a node may contain a sub diagram describing the state represented by that node in more detail. A state node in a diagram represents a state within the collaboration protocol. A link in a diagram represents an event. The occurrence of an event causes a transition between the two states connected by the link. Nodes are identified by a unique name within a collaboration protocol.

Each state of the collaboration is associated with a set of rules. A rule specifies which functional-roles are permitted to perform which operations on what type of artifact. A functional-role refers to a collection of privileges and responsibilities in a given type of session. E.g., a reviewer is a kind of functional-role whose members can comment on design artifacts in the state of reviewing and who are not allowed to delete the designed artifacts.

A link will be bound to an action that is represented by executable code. In addition, who can perform the action will be also specified as a property of the link. A link can be bound to a conditional expression as well.

There are four additional types of nodes. They are or-split, and-split, or-join, and and-join nodes, respectively. When an or-split node is active, the node will trigger one of its output events depending on the result of evaluating the conditional expression that is bound to the node. When any input event of an or-join node occurs, it will trigger the output event. An and-split node allows all its output events
to occur in parallel. An and-join link triggers its output event only if all of its input events occurred.

There are two other types of nodes: default node and exit node. If a state node contains a sub diagram then there is exactly one default node defined in the sub diagram, which will be activated when the containing state node becomes active. Subsequently, all the state nodes that are linked to the default node will be activated. An exit node is a logical pointer to a state node that is not located in the current diagram. When an exit node is activated, the system will search for the collaboration state node that has the same name as the exit node, and then the state node found becomes active. The advantage of this formal modeling language is that it has all properties of statecharts and it is not necessary to represent the whole process in one complex diagram.

3.1.4 Contract

All participants of a session will play one or more functional-roles in the group interaction process. A functional-role's members are specified by contracts with members, organizational-roles or organizational-units during the specification of the session. Its membership can be determined dynamically by the properties of team members as well. Using the concepts of functional-role and contract can be beneficial. The definition of functional-roles is behavior-oriented, while organizational-roles or organizational-units are defined in an organizational context. A participant with a certain organizational-role or being a member of an organizational-unit may behave in quite different manners in different interaction processes. For example, a project manager takes the role of a mediator in a project management meeting while he is a reviewer in a technical meeting. Changing the definition of functional-roles has no influence on the definition of organizational structure, and vice versa.

The process of defining group behaviors is carried out when specifying a session.

3.2 Support for Group Interaction at run time

In this section we describe how to support group interactions at run time. Here, the concept of dynamic binding plays an important role.

3.2.1 Support for Group Interaction Processes

Group interaction processes refer to the processes that take place when team members actually interact following an agreed collaboration protocol. In this paper, we focus on indirect group communication and interaction by manipulating shared artifacts in a session, rather than direct communication and interaction by using audio or video tools.

All authorized participants can join or leave a session freely. When he/she joins a session, the top-level of the shared information space of the session is displayed in the session browser. He/she can browse through the artifacts following the links. Links represent dependencies between artifacts. In this browser there is a palette which lists those of the object types of the currently used activity space, which are allowed in the current state according to the rules defined in the collaboration protocol of this session. Figure 1 presents an example of an activity space for software requirement specification, which is presented in a session browser. The browser shows a window title bar listing the document name and tool type. Below it is a list of icons showing the current users of the tool (e.g., Haake and Miao). From left to right, the window contains a button bar with generic functions for editing, navigating and process control, followed by a palette of possible node and link types (e.g. “requirement” and “is-a-superfunction”) and actions (e.g. “make decision” and “continue”) in this activity space, followed by the content window displaying the activity space content.

Multiple participants can work in the same session simultaneously. The shared artifacts can be viewed and edited simultaneously by authorized team members working in the session. When a user performs an operation on an artifact in the information space of a session, the system processes it as a transaction. First, the system will check whether this member is permitted to execute the operation according to the rules defined for this state of collaboration. If this check fails, nothing happens except for displaying a warning message. If the check is successful, the transaction will be propagated to all sites to keep consistence. When he/she enters a sub-activity space by opening an artifact node, the system will display the content page of the node in the session browser. Meanwhile, the system will update the palette with a list of predefined object types that are allowed to be instantiated in this sub-activity space (as defined by the type of node/activity space).
The system provides both manual and automatic transitions between states. Currently allowed actions are also listed in the palette of the session browser as action buttons. The names of these buttons are the names of the corresponding links in the collaboration protocol. In manual mode, when users with appropriate functional-roles click an action button, the corresponding event is created, which triggers the assigned action to be executed. As a result, a state transition will be caused according to the definition of the used collaboration protocol. In automatic mode, an autonomous agent (background process) monitors the status of the collaboration periodically by evaluating the transition conditions and changing the state accordingly when a condition has been met. When changing the state of collaboration, the system will update the palette in the session browser with the list of artifact types and action buttons that can be used in the new state.

3.2.2 Dynamic Binding

Users of SCOPE have a wide range of flexibility to change the definition of group interaction dynamically without recompiling the system. They can change the organizational model, design models, collaboration protocol, and contracts. The system always uses the newest version of these definitions. Now we discuss two important dynamic aspects of SCOPE.

Dynamic Functional-Role

In some cases, the membership of a functional-role can not be determined by contracts with certain members, organizational-roles or organizational-units. It has to be decided considering the properties of members and situational dynamics. That is, a given operation or action must be performed under certain conditions by a person who is qualified. An example for such a condition is that the state of a given artifact is "proposed". Examples for qualified people are people who are of a given specialty, the owner of a given design artifact, or participants of a given session. SCOPE enables end-users to describe the dynamic functional-role by selecting a suitable predefined predicate and defining the actual parameters. The membership of a functional-role will thus be determined at run time.

Dynamical Change of a Collaboration Protocol

During group interaction, the collaboration policies may have to be adapted to fit some change. Here we give an example to illustrate the adaptation process.

A team of three members, two engineers and a client, works in different places to collect and analyze the requirements for developing an application software. In the end, they have to work out a requirement specification and deliver it. Without considering coordination with other teams (e.g., system design team), they have at least to coordinate their own efforts to get the work done. First, they work together in a session following the collaboration protocol described in Fig. 2. Note, the collaboration protocol editor in Fig. 2 shows a similar user interface to an activity space browser. Only the generic button bar (now including protocol specific editing functions) is located below the user list. This collaboration protocol consists of five nodes which represent five top-level states, respectively, defined, enabled, active, suspended, finished, and links which represent transitions. The popup window shown also in the Fig. 1 is used to set permissions in the state “active”. When the current state is “active”, all of them can join/leave the session at any time and perform any operations in the shared workspace of the session. They work in a loosely coupled collaboration mode. That is, each member can browse individually in the shared workspace and create/delete/read/modify any requirement/comment. In this collaboration protocol, there are two kinds of functional-roles: requirement analyst and leader. Only the leader has the right to perform the "deliver" action. At first, this collaboration policy works well for a relatively small number of members and at the preliminary phase.

As work progresses, they find that the problem is not as easy as expected and the task is hard to finish in time. Then, the number of team members is increased. They have to adopt a more formal collaboration policy to fit this change. Especially, one of the new members, an experienced senior analyst, does not work in this team full-time. However, the requirement specification has to be approved by him before being delivered. Thus, the collaboration protocol is modified by adding a sub-diagram in the content page of the "active" node and by setting the access rights appropriately (see Fig. 3).
is defined. In this sub diagram, there is a default node (dot) and an exit node “suspended”. At the state of "collecting", all team members have the right to create new requirement nodes and fill the values of information units in the corresponding requirement node. It is allowed for users to view the content of requirement nodes created by others. However, it is not allowed to delete requirement nodes except for the nodes created by the same user. Only designated members (e.g., having the functional-role "leader") have the right to perform the action "meet". This action results in a transition to the state "discussing" and changes the collaboration mode from loosely-coupled to tightly-coupled. In this state, all participants have the same view in the content window of their session browsers and navigate together. All team members have the right to create nodes/links and modify values of any artifact nodes. They may use a floor control policy to coordinate their synchronous activities.

The floor control policy can be described as a sub diagram in the content page of the node "discussing". Because of space limitations we don't discuss this process thread further. Some designated members (e.g., leader, floor holder or senior requirement analyst) can perform operations such as "delete a node or a link" which were created by others. The leader has the rights to perform the actions "make decision" and "continue" (see Fig. 1).

Different actions performed result in different transitions. If the action "make decision" is performed, action buttons "vote for", "vote against" and "abandon" will appear in the palette of the session browser. All members can click any one of them once. The link "all voted?" is bound to a conditional expression. In the state of "voting", the system will evaluate this expression periodically. After all members voted, the system will evaluate the conditional expression assigned to the or-split node "all agree?" and will change the state according to the result of this evaluation. If all members agree to the current version of the requirement specification, the state will become “reviewing”. In this state, the experienced senior analyst can decide whether to deliver it or not. If yes, when the button “deliver” is clicked, the requirement specification will be transferred automatically to other sessions according to the overall process definition. Meanwhile, the state of collaboration of this session will change to “suspended” indicated by the exit.

4 Comparison to Related Work

In order to control group interactions in a shared workspace, mechanisms can be provided at the data level or at the process level. Mechanisms provided at only one level are not sufficient. Some systems allow users to set access rights. For example, Shen et al. [9] presented an access control model to meet the requirements of collaborative environments. By means of setting access right explicitly, some unpredictable events could be avoided, but the users would bear a heavy burden when changing access right settings for different collaboration states during collaboration.

Process control mechanisms are provided in some collaborative design support systems. For example, PRO-ART [6] supports synchronous and asynchronous collaboration at a phase of a concurrent engineering process. However, there is no access control. All participants have all authorities to access any artifact.

Some collaborative design support systems control group interaction by exploiting coordination mechanisms at both levels. However, the access control tends to be an all-or-nothing affair at a given state for a participant. For example, in the system described in [3], the workflow model consists of three work states: project management, artifact revision, and artifact review. Members with different responsibilities are assigned to each work state. Although members assigned to different work states have different authorities to deal with artifacts, members working in the same state of work have the same set of rights to manipulate the shared artifact which they are designing.

The other issue addressed in this paper is dynamic binding. For some systems it is not necessary, because the exploited coordination mechanisms are general-purpose and impossible to combine with application software. For example, floor control is a mechanism that can be used in desktop conferencing during collaborative design processes. When a participant has the floor token, only he/she has the access rights to access artifacts and others are can do nothing to shared artifacts except for "observing". In SCOPE, floor control policies that are personalized can also be defined as part of collaboration protocols. Beyond that, in SCOPE the floor holder is regarded as a dynamic functional-role which may have specific access rights, while other participants
not being floor holders may have some non-conflicting access rights at the same time. This provides more flexibility.

Some collaborative design support systems are designed to bind group interaction semantics to functionality of design support software such as CAD. An example of such systems is described in [3]. This system implements a work flow model in which work state, roles, frames of design documents and procedures are defined at the system development phase. In this process, members with different roles should be assigned to each work state. Users of this system have to interact according to the build-in collaboration policy. Only recompiling the system can change the collaboration policy.

Binding at the system development phase relieves the system’s users from the burden of defining group interactions. It is more efficient and it should be simple to be used. For well-structured design processes this approach is advisable. However, design is a creative task and collaborative design processes are normally ill-structured, long-lived, and situated. Collaborative design support systems should therefore enable end-users to build collaboration policies at run time.

5 Conclusions

Collaborative design systems support cross-functional teams in performing design work collaboratively. However, when team members only use social protocols to coordinate their activities, potentially unexpected interactions and unpredictable conflicts may occur when manipulating shared design artifacts. Therefore, there is a need to provide a computational mechanism to support group interactions in the shared workspace. Such a computational mechanism should be (1) incorporated in the design support software, (2) accessible to the team members and (3) modifiable at run time.

Our approach is to provide a mechanism to control the interaction of teams by combining data level control mechanisms and process level control mechanisms, and to dynamically bind group interaction semantics to the properties of a collaborative design software system.

We focus on support for individual cross-functional team work in a shared workspace. SCOPE supports a team to work in different styles in different situations. They can perform activities synchronously or asynchronously; individually or jointly; and routinely or spontaneously. The system provides flexibility for end-users to dynamically change design models, organizational structure, interaction procedures, functional-roles — in short: group interaction semantics. By combining the data level control mechanisms, such as access control and concurrency control, with process level control mechanisms, group interaction can be controlled in a fine-grained and systematic manner.

First experiences with SCOPE have shown promising results. By using a direct manipulation interface end-users can define their own collaboration protocols without any coding. However, end-users may define and use incorrect collaboration protocols. Thus, checking functionality should be provided by the system. E.g. grammar errors may be easy to detect, such as isolated state errors. In some cases, conventions can be used as clues to check for semantic errors. As a next step, we will test and evaluate the usefulness of the system in more real-world settings.

References