Supporting Concurrent Design by Integrating Information Sharing and Activity Synchronization

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
Viewing concurrent design as processes of collaboration, coordination and co-decision making within and between cross-functional teams, we argue that an appropriate integration of information sharing and activity synchronization approaches is necessary and beneficial for a concurrent design support environment. This paper presents our approach to model concurrent design processes. We address support of information sharing and activity synchronization within a team and between teams. The prototype system SCOPE integrates process support technologies into a cooperative work environment. It provides support for specification, modification monitoring, and execution of session-based collaborative processes.

1 Introduction
The traditional procedure in industry for designing a product is the implementation of sequential design, often referred to as the „over-the-wall” approach. In recent years, Concurrent Engineering (CE) has become a widely accepted concept and is regarded as an excellent alternative approach to the sequential engineering process. CE is a systematic approach to integrated product development that emphasizes response to customer expectations and embodies team values of cooperation, trust, and sharing in such a manner that decision making proceeds with large intervals of parallel work synchronized by comparatively brief exchanges to produce consensus [3]. The objective of CE is to reduce the product development cycle time through a better integration of activities and processes. Parallelism is the prime concept in reducing design lead-time and concurrent engineering becomes the central issue. In this research we focus on the early phases of this process and therefore deal only with concurrent engineering concepts for the design phase of products.

Concurrent design involves the interaction within and between diverse cross-functional teams of individuals who may be scattered over a wide geographic range. Within and between such teams, collaboration, coordination and co-decision making are critical to a successful design. The main technique for promoting collaboration and co-decision making between team members is to provide information sharing facilities, whereas the main technique for coordinating work is to provide facilities for synchronizing the various activities. Adopting an information sharing approach, most of existing environments, e.g. SHARE [22], CoConut [11] and CASCADE [1] can help to bring disciplines and domain experts together quickly so that they can pool their knowledge and work effectively, irrespective of location or time. However, such systems provide less support for users to organize and coordinate activities in order to achieve a goal in an organizational context. Coordination relies to a strong degree upon the experience and interaction between individuals and teams. Workflow, which addresses activity synchronization, is a technology that supports complex processes. Some studies [2, 12] introduce workflow into CE as an important technology to support coordination. However, current Workflow Management Systems (WFMS) do not have adequate support to satisfy the modeling and correctness requirements of concurrent design processes. The deficiencies of WFMS include lack of support for synchronous collaborative activities and for interactions between concurrent activities. Obviously, strengths and weaknesses of information sharing and activity synchronization approaches are complementary. Hence, concurrent design support environments would benefit from appropriate integration of information sharing and activity synchronization approaches.

This paper investigates support for defining a set of coordinated and concurrent collaborative activities that constitute a design process in the form of a process model, and for executing, monitoring and dynamically modifying process models at runtime. We propose two key concepts: „session” and „session-based collaborative process”. The concept of a „session” is defined here as that in a period of
time team members perform design activities in a shared information space. The concept of a „session-based collaborative process” is defined as a set of coordinated and potentially concurrent sessions. The prototype system described in this paper, SCOPE (for „Session-based Colaborative Process-centered Environment”), integrates a process modeling approach into an information sharing environment. Our focus is on combining information sharing and activity synchronization to support concurrent design activities.

This rest of this paper is structured as follows: In the next section, characteristics of concurrent design processes are analyzed and major requirements are identified. Section 3 presents related work. Section 4 introduces our approach to combine information sharing and activity synchronization. Section 5 provides our conclusions.

2 Characteristics of concurrent design processes and main requirements

In this section, we investigate concurrent design processes from three important perspectives: collaboration, coordination and co-decision making. In order to build a support environment for concurrent design, some requirements for supporting collaboration, coordination and co-decision making are identified. Such support is addressed here in terms of information sharing and activity synchronization dimensions.

2.1 Concurrent design as a collaborative process

Collaboration is defined here as a process where individuals share a common goal and need to work together to achieve it. At the end of the collaborative process, the single contribution of individuals can not be isolated because the product is an entity that unifies the results of all individual contributions of the cooperative ensemble. Team design is viewed as a social construction of a technical reality focused on establishing and maintaining a „shared understanding” among the participants [22]. Designing an artifact is a complex knowledge intensive process carried out by a group of heterogeneous participants working from different perspectives [1]. Integration of knowledge among participants is obtained by collaborative idea generation and development through discussions and brainstorming etc. Curtis et al. [4] found that negotiations about requirements and trade off of solutions occur throughout the design processes. Reddy et al. [18] point out that information sharing involves developing common data representations and providing transparent access to information. From above, the following two requirements can be identified:

(R1) Enable teams to build their design models jointly.

The information must be organized into a framework that represents the evolving design. This representational framework is called a design model. The design model forms a basis for knowledge sharing among team members. Team members perceive the goals and the design problems differently depending on their knowledge and interests. The establishment of a design model is a prerequisite for communication and for the interpretation and integration of information. This is especially true for multi-disciplinary design teams where much valuable information is cross-disciplinary. Particularly during early phases teams spend considerable time defining terminology and common views. A design model is usually obtained by having a series of meetings where each participant describe his/her view on the problem.

(R2) Control access to the shared information space.

Through the evolution of the design process, various models of the product will exist and interact at various stages. Different types of models will exist at the same stage, e.g. geometrical, behavioral, etc. Different disciplines and domain experts have different rights to access and to modify various models or information units at different states of the design process or the artifacts.

2.2 Concurrent design as a coordination process

Coordination refers to a process where individuals or teams need to coordinate their actions with those of others. The main problem in coordination is the synchronization of people, actions and the consistency of the individual actions with respect to the whole process. A key factor in the success or failure of a multi-disciplinary, multi-team design activity is the management and control of the various design activities. Different activities are being carried out at different times for different purposes. They need to provide certain information at particular times [6]. Ellis [8] distinguished coordination at two levels: at the activity-level and at the object level. Based on this distinction two more requirements can be identified:

(R3) Support activity-level coordination.

Activity-level coordination refers to the synchronization of activities that make up a procedure. Polkrock et al. [16] found that product development is a highly organized form of knowledge work. Teams are involved in a continuous cycle of planning, implementing, monitoring, and improving the collection of activities vital to the success of a given project. The relationship of two activities may be sequential, parallel or concurrent. Parallel activities refer to two or more simultaneously active activities, which deal with different artifacts. Concurrent activities refer to two or more simultaneously active activities, between which artifact interdependency
exists. An activity may be assigned to an individual or a team. Therefore, facilities are needed to support specification, modification, monitoring, and execution of design processes. Especially, concurrent activities performed by different teams or by individuals in the same team must be synchronized.

(R4) Support object-level coordination.

Object-level coordination is concerned with how sequential or simultaneous access to the same objects is managed. For example, how to deal with two members’ requests to modify the same object at the same time. Solutions like locking or floor control are not suitable ways of object-level coordination for concurrent engineering because they hinder synergy. In order to facilitate close collaboration and synergy in design processes, systems should enable participants who carry out different tasks to access the same set of artifacts simultaneously and to prevent inconsistent changes. Every change must be propagated automatically so that the version of the artifacts accessed by any member must be up-to-date at any time. Thus, more complex concurrency control is required.

2.3 Concurrent design as a co-decision making process

Effective, accurate, and timely decision making are essential for successful performance of CE [7]. Through the course of design problem solving, teams make multiple decisions as to the best way to achieve the design objectives and to utilize the available resources. Thus, there are needs to:

(R5) Support sharing information related to decisions.

When several people participate in a decision process, information sharing has a prominent role, as in collaboration in general. Decisions made by the designer will require a large volume of design information. The design process itself will generate more design information. This information must be shared with affected personnel [9]. Systems must reduce the effort of decision makers to share all information useful for the decision to be taken, such as decision criteria and already taken decisions, in order to avoid to continuously return to the steps already made and to leave some participants out of the process.

(R6) Support structuring co-decision making process models.

In co-decision making, consensus seeking, voting and negotiating highlight the activities that team members engage in. Allocation and sharing of responsibilities are the prerequisites for co-decision making. Different teams may adopt different strategies or policies to reach a decision. Systems should support team members to define and customize co-decision making process models and to coordinate activities embedded in the processes. Systems must help to reach a good concurrent engineering decision by transferring information quickly, easily, and clearly among people who perform different activities [7].

In summary, collaboration, coordination and co-decision making are critical to a successful design. In order to support these three activities, information sharing and activity synchronization should be integrated into a comprehensive concurrent design support environment.

3 Related work

Most of the existing environments for CE are developed by using information sharing approaches, such as Flecse [5], CASCADE [1], CEDTS [9], OOWB [7] and CoConut [11]. These systems allow a group of designers to concurrently view and even edit shared artifacts. This systems are so called single stage systems according to the definition in [8]. When a group of designers are working on a complex design, the primary means of interaction for them is to interact through the design artifacts. These systems act primarily as a repository for and a controller of access to the artifacts. The collaboration model does not provide any temporal sequencing of activities. Every group member can perform any activities in the shared information space at any time. Any modification is propagated to other group members. These systems have complex object-level coordination and a trivially simple activity level coordination. Most systems in this category can support object-level coordination and in different degrees (R4), and some of them enable team members to build their design models jointly (R1), control access to the shared information space (R2), and support sharing of information related to decisions (R5). However, these systems provide less support for activity-level coordination (R3) and for structuring co-decision making process models (R6).

Some information sharing systems add means for structuring the group’s overall work process to a general collaboration environment. For example, the system described in [15] is based on a model of three-phase engineering processes. Within the first phase, people work in parallel and use different products to represent their views. In the second phase, the products (views) are interrelated as the result of conversations and collaboration. In the third phase, people work in parallel again, but unlike the first phase, existing interrelations must be considered and maintained. Then an arbitrary numbers of loops, consisting of second and third phases, will be performed until the final product fulfills the expectations. It should be noted that this system does not have a process enactment engine. Obviously, it provides less support to model processes in terms of activities and artifacts.
Some studies [2, 12] introduce workflow into CE as an important technology to support activity-level coordination. However, current WFMS do not provide adequate support to satisfy the modeling and correctness requirements of concurrent design processes. Normally, current workflow systems prevent users from editing a design module until another user is finished with that design. In these systems, interactions between concurrent activities can cause inconsistencies, and support for synchronization of concurrent activities is weak. That is, these systems support activity-level coordination (R3) and structuring co-decision making processes (R6), but offer less support for team members to build their design models jointly (R1) and for object-level coordination (R4).

Some recently developed systems in the workflow community, such as WoTel [24], integrate synchronous collaboration tools into workflow systems by supporting meeting-like steps in the workflow. Embedding synchronous teamwork as part of the workflow means a change in the workflow paradigm which was limited to „one person - one task - one application”. This integrated approach is also used to develop systems for CE. E.g., the system described in [13] exploits a conference method in each stage of concurrent engineering for appropriately integrating and exchanging information between a real-time conference system and an asynchronous cooperative system. Such systems enable a continuous stream of activities in which fast, informal, ad-hoc, and direct actions can be taken through distributed meetings within the usual formal workflow. These systems can deal with the hand-over of documents and the control flow between meeting-like steps and normal steps. That is, after meetings the documents of the meetings flow as minutes into the succeeding steps. Other artifacts produced in the meeting process disappear as the meeting ended. Therefore, the scope of collaborative activities are limited exactly to single steps of a workflow. In addition, in these systems artifacts can not be accessed simultaneously by users working at different steps of a workflow, which is required by concurrent design situations.

In summary, most existing systems for CE are developed by adopting primarily one of approaches and support only part of the requirements identified in the previous section. Although some information sharing systems are developed with some basic process support functionality and some workflow systems for CE are developed by integrating synchronous collaboration tools, there is still no comprehensive environment for concurrent design which can meet all requirements identified in the previous section. In the next section, we will present our integrated approach to build such an environment.

4 The SCOPE system

As discussed above, in order to build an environment to support collaboration, coordination and co-decision making in concurrent design processes, it is necessary to find an appropriate approach to integrate information sharing and activity synchronization. In this section, we present our approach to support concurrent design processes and focus on discussing how to support information sharing and activity synchronization within and between sessions.

4.1 Sessions

Concurrency is the key for CE. Parallel work in cross-functional teams represents one of its most important enablers. A concurrent engineering environment must support several distinct parallel teams that specialize in a variety of disciplines. Inside each team, members communicate with each other closely and work together on creative solutions. The overall design work should be divided into sizable chunks of semi-independent activities which are assigned to the individual teams so that each one can work in parallel [17].

The concept of a „session” is defined here as that in a period of time team members perform activities in a shared information space. A session is created for a cross-functional team to solve a long-lived 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. The artifacts are persistent, that is, the artifacts continue to exist even when all members leave the session, and when one or more members join the session again they will find the artifacts unchanged. Team members can perform synchronous or asynchronous activities in a session to construct shared artifacts together. These activities may be performed following a predefined process or as an ad-hoc process according to different teams or different situations. That is, team members may perform activities individually or jointly, and routinely or spontaneously. Furthermore, 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. They have to coordinate their efforts to get their work done.

The process of specifying a new session begins with the definition of a meta-specification object for the session. To define a new session users are required to define attributes for the session meta-specification. Without being exhaustive these attributes include: session name, session type, session creator, session description, collaborative mode, activity space, collaboration protocol, mapping
from organizational to functional roles, scheduled start and end times, deadline, etc. A discussion of two more important attributes is presented below.

4.1.1 Activity spaces

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 [21]. Using a labeled graph representation, it can be regarded as a semantic network which has structural, relational, and computational semantics [23]. For example, a product design model or an argumentation model can be defined in an activity space which can be instantiated to capture specific design or argumentation structures. 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 COWFISH system [23] extends the SEPIA [10, 21] activity space concept into a meta-model for defining hypermedia-based activity spaces. The elements of the meta-model are nodes, links, node content pages, and other media objects such as text, drawing, graphics, image, source code, 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. Based on the meta-model, a set of tools have been developed to help users define, manage, use, and create activity spaces in COWFISH. These tools enable product designers to define their own product models easily. One of these tools, called the flexible space, allows 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. Concurrently, information is abstracted and reorganized so that it can be shared by team members. The tools of COWFISH have been integrated into SCOPE. Defining the attribute „activity space” for a session in SCOPE means choosing a type of activity space as the root activity space of the shared information space of the session.

Fig. 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 and actions (e.g., „decision”) in this activity space, followed by the content window displaying the activity space content. The requirements for the specification of some software are modeled in the activity space as a Directed Acyclic Graph (DAG) of requirement elements. Each node in the graph is used to represent various aspects of a functional requirement and is identified by node name. Each link represents an is-a-subfunction relationship. The use of a DAG enables relationships among requirement elements to be structured, and it also allows multiple requirement nodes to share a sub-node representing a common sub-functionality. The content page of a requirement is accessible via the requirements editor and contains a set of information units such as name, state, owner, description.

4.1.2 Collaboration protocol

A collaboration protocol is used to coordinate activities in a session. It is described as a state-transition diagram where a node may contain a subdiagram describing the object’s possible behavior in more detail while it is in the corresponding state represented by that node. The different nodes in a diagram represent different states within the collaboration protocol. SCOPE provides both manual and automatic transitions between states. In manual mode, team members with appropriate functional-roles perform an action to trigger the transition between states. In automatic mode, an autonomous agent (background process) monitors the status of the collaboration periodically by evaluating the transition conditions and changes the state accordingly when a condition has been met. In each type of session, there is a collection of domain-specific actions which are represented by
executable code. Actions take place within the context of structural conditions which are able to change over time and/or in response to contingencies. Each state of the collaboration is associated with a set of rules. A rule specifies which functional-roles are permitted to perform which actions or which operations on artifacts, and which valid actions result in state transitions. 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 the artifacts in the state of „reviewing”. A functional-role’s membership is determined by contracts with organizational-roles during the specification of the session. An organizational-role refers to one person or a group of people in an organizational context, such as department manager. Using the attribute „Collaboration protocol” a collaboration protocol is specified for a session. In SCOPE, it is allowed for team members to modify a collaboration protocol on the fly.

As an example of a collaboration protocol, a protocol for specifying requirements is given in Fig. 2. Note, the 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 (see upper window in Fig. 2). A state may contain a sub diagram, e.g. the state „active” contains a sub diagram (see lower window in Fig. 2). The sub diagram contains four states of the collaboration: collecting requirements, analyzing & organizing requirements, reviewing requirements, and voting. Within a node, a set of behavior rules is defined. At node „collecting”, for example, 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 mediator role) have the right to perform the action „analysis”. This action results in a transition to the state „organizing”. In this state, all team members have the right to create nodes/links and modify values of any nodes. Some designated members (e.g., a mediator or senior requirement analysts) can perform operations such as „delete a node or a link” which were created by others.

4.2 Session-based collaborative processes

Most phase models present the product development process as a serial chain of activities, such as the descriptive model of Pahl and Beitz [14]. Implementing concurrent engineering implies restructuring the product development process. The order of activities has to be reconsidered so that some activities can be performed in parallel. The interactions between the various activities have to be defined in order to provide the necessary consensus and a check of the integrity of the product [19].

In SCOPE, the term “session-based collaborative process” (SCP) refers to a set of coordinated and potentially concurrent sessions used by team members to achieve a goal. In order to enable activities to be performed concurrently, a methodology is needed to model more complex relationship between activities. In this sub-section, we discuss how to support definition and execution of a session-based collaborative process.

4.2.1 The Process model

Every SCP is represented by its description, which is called a process model. This description specifies all sessions embedded in the process, the values of attributes of each session, and the relationships among these sessions. SCOPE provides a visual process model language for process definition. A process model is described as a hypermedia document consisting of layered nodes which are connected via links. A process can potentially be decomposed. A process or a sub-process is represented visually by a rectangle with the label „process” and the name of the process. The components and structure of a (sub)process are described on the content page of the node. A session is represented visually also by a rectangle but with the label „Session Node” and the name of the session. Session nodes are elements of processes and each session has a number of attributes which have to be specified. A relationship among sessions and sub-processes is represented as a typed link in the hypermedia document. A process model is independent of
a particular use situation, but it may be customized and instantiated to produce a particular collaboration design process to fit the needs of a particular project. The defined process models can be stored in a process model base. However, what should be described in a particular process model, and to what level of detail, should be determined by the needs of the team members. Every process execution is represented by a process instance, which evolves according to information included in the corresponding process model. A process, which is described by a single process model, may have an arbitrary number of instances. Process instances and process execution will be discussed in section 4.3.

Next, we discuss two kinds of relationships between sessions: temporal relationship and artifact relationship.

4.2.2 Temporal relationships

A temporal relationship represents the time dependence between sessions, such as precede and delayed-start. If a session A precedes a session B, it means that when A is finished, the active-condition of B is evaluated. If a session A delayed-start a session B with a variable “interval time”, it means that when A is finished, after the “interval time” delay is over, the active-condition of the B is evaluated. SCOPE provides four types of more complicated temporal links. They are or-split, and-split, or-join, and and-join links, respectively. Split links allow to fork parallel activities and join links provide a simple mechanism to synchronize activities. They are used to specify when the execution can continue: either all of the preceding sessions are finished or at least one of them needs to be finished.

4.2.3 Artifact relationships

In order to address the problem of information overflow for a team, not all artifacts about a project should be included in the shared workspace of a session. Only artifacts required and produced by the team are placed in the shared workspace so that team members can focus on interesting artifacts. However, interdependence among artifacts placed in the workspaces of different sessions may exist. Some artifacts should be viewed and even manipulated synchronously or asynchronously by the people working in different sessions, and some artifacts are transferred from one session to another after certain work steps. An artifact relationship is used to represent a kind of dependence of artifacts between sessions, such as transferring and sharing. An artifact transferring relationship denotes the situation when an output artifact of one session will be transferred into another session as an input artifact. It is represented as a directed link which is shown in Fig. 3 as an arrow with a black handle pointing to/from a named rectangle representing an artifact buffer.

![Figure 3: Description of a partial software design process](image)

An artifact sharing relationship means that an artifact can be viewed and manipulated by people working in different sessions running concurrently. This kind of relationship is represented as a bi-directional link. It is important to note that parallelism of work requires some coordination of the artifacts produced or consumed by activities running in parallel. Unlike current workflow systems, in SCOPE the artifact flow does not necessary follow the ordering imposed by the control flow. It is allowed to define two artifact flows in opposite direction between sessions. As shown in Fig. 3, flows of artifacts in two directions can co-exist between the sessions „Identifying classes/objects “ and the session „Identifying semantics of classes/objects”. Furthermore, transferring artifact and sharing artifact relationships can be used to control and even synchronize concurrent sessions. We will discuss this mechanism next.

4.3 Process execution

Process execution is concerned with the enactment of a process following the previously defined process model. In general, processes are interactive and require intensive information exchange with team members. Now, we discuss process execution from the users’ perspective.

Empowered team members can select a pre-defined process model from the process model base as their initial process model. If no suitable process model exists, they can simply define a new one. This process model serves as an overall design plan. When they start to execute the process model, a so called process instance of the process model is created. This process instance will progress according to its process model. As the process instance progresses, the state of the sessions described in the process model will change. Even when a process instance has already been created, parts of the process model could be modified. That is, the values of attributes of the corresponding sessions could be modified and the model of a sub-process could be altered. The process model of a sub-process acts as a sub-plan. It should be noted that
defining or revising a plan itself is carried out in a session. In particular, the session to define the overall plan lasts until the end of the overall process. Thus, team members with appropriate roles can monitor or change the plan.

Each team member has a personal information panel. In this panel, information is listed about processes and sessions in which he/she should take part. Team members can get more detailed information about sessions through the list, such as state, description, scheduled start time and estimated duration, participant list, and so on. He/she can join a session by selecting it from the session list. When he/she joins a session, the top-level of the shared information space of the session is displayed in the session browser and he/she can browse through the artifacts following the recorded dependencies. In this browser there is a palette which lists those of the object types of the currently used activity space (which defines the shared information space, cf. 4.1) which are allowed in the current state according to rules defined in the collaboration protocol of this session. Each team member can be a member of more than one session concurrently and 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 rules defined at 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 a node, the system will display the content page of the node in the session browser and update the palette with a list of pre-defined object types which are allowed to be instantiated in this sub-activity space (as defined by the type of node/activity space). Transitions from one state of collaboration to another may be carried out by team members who have the right to perform actions. Allowed actions are also listed in a palette in the session browser. State transitions may be carried out automatically, too. Each session has an autonomous agent which monitors the status of collaboration periodically by examining the transition conditions. If a condition has been met, the system will change the state according to the collaboration protocol. In Fig. 2, for example, when the current state of collaboration is „voting“, and all team members have voted, the current state of collaboration will become „organizing“ or „suspended“ according to the result of the voting. No matter whether a transition is carried out manually or automatically, all team members on line will receive a notification about the change of the state of collaboration. When changing the state of collaboration, the system will update the palette in the session browser with a list of allowed actions (e.g. see Fig. 1 „decision“) that can be performed in the new state.

Changes of the collaboration state of one session may result in changes of state of related sessions according to the process model. In the case of two sessions connected by a temporal link, once the source session is finished, the state of destination session will change according to the process description. In SCPs, artifacts can also be used to control execution of processes. SCOPE provides „artifact-trigger“ mechanism to control execution of processes. For example, in Fig. 3 when the first version of requirement specification produced in the session „Specifying requirements“ has been delivered, this mechanism allows the session „Identifying classes/objects“ to become active, although the sessions „Specifying requirements“ has not yet terminated. If we would like to allow the two sessions to proceed concurrently, but in a coordinated fashion, we can also use the „artifact-trigger“ mechanism. For example (see Fig. 2 and Fig. 3), after a new version of requirement specification is released in the session „Specifying requirements“, the state of collaboration of this session becomes „suspended“. Once the comments (e.g., inconsistency among requirements) are created in the shared artifact „requirement specification“ by the people working in the session „Identifying classes/objects“, the collaboration state of the session „Specifying requirements“ changes into state „active“ and sub-state „collecting“ as default. Meanwhile, when the session „Identifying classes/objects“ received a new version of requirement specification, the participants of this session will browse the new requirements and may comment on it. They define candidate classes/objects. After they review and decide to deliver their results, a new version of „candidate of classes/objects“ is released. At same time they wait for feedback (e.g., inconsistencies in candidate classes/objects) from session „Identifying the semantics of Classes/Objects“ as well. Whenever the transferred artifacts arrive, they may result in a state change of the session according to the collaboration protocol. Using this mechanism, changes in artifacts can lead to suspension or re-activation of long-lived tasks. Therefore, coordination of concurrent sessions can be achieved by controlling and synchronizing activities via shared artifacts.

4.4 Implementation issues

The architecture of the SCOPE cooperative design environment is shown in Fig. 4. This environment is implemented based on our COAST toolkit [20] in VisualWorks Smalltalk. The COAST toolkit offers many collaboration support features like replicated shared objects, concurrency control, and pre-defined as well as
Software components in the kernel include the enactment and access control components. The enactment component interprets the process models, monitor and control process instances and session instances. The access control component acts as a filter of users’ actions and operations. It should be noted that defined activity spaces, collaboration protocols, process instances and session instances, and product data will be referred by the access control and the enactment components.

5 Conclusions and future work

In SCOPE, concurrent design is viewed as processes of collaboration, coordination and co-decision making within and between cross-functional teams. We elicited the major requirements for supporting these three activities within concurrent design processes. Information sharing and activity synchronization are two key approaches which have to be integrated into a concurrent design support environment. Based on the concept of “session-based collaborative processes”, we developed a modeling methodology and tools which enable team members to model concurrent design processes, and we built a prototypical cooperation environment to support modification, monitoring, and execution of session-based collaborative processes.

Specifically, we focus on support for individual cross-functional teams which allows them to work in different styles in different situations. They can perform activities synchronously or asynchronously; in the same place or in different places; individually or jointly; and routinely or spontaneously. This is realized by exploiting COAST’s facilities to maintain a shared workspace with the aid of data level control mechanisms, access control and fully replicated concurrency control, and user level control mechanisms, i.e. collaboration protocols.

Equally, we emphasize support for cooperation between cross-functional teams. Because there exist complicated interdependencies between activities carried out by different teams, facilities are provided so that people working in different teams can share artifacts synchronously or asynchronously. Furthermore, the shared artifacts can be used to synchronize not only sequential but also concurrent activities carried out by parallel teams.

Most systems for CE are developed by adopting either activity synchronization approaches or information sharing approaches. These systems usually emphasize either the aspect of collaboration or the aspect of coordination. Therefore, they provide insufficient support required by collaboration, coordination and co-decision making. Some systems are developed by adopting an integrated approach in which some general synchronous collaboration tools are integrated as external tools into a process-centered
environment so that team members can perform a meeting-like activity as a stage of a process. Our system is developed by integrating process support technologies into an information sharing environment. The SCOPE system can support synchronous and asynchronous cooperation among people not only within a team but also in different teams. Furthermore, the system provides a high level of adaptability and flexibility for team members to model design models and concurrent design processes. Hypertext technology is used due to its inherent flexibility for information structuring as a basis for information processing and process support.

Generally speaking, too little synchronization will result in duplicated or vain efforts, while too much synchronization will lead to over-formal communication between its members. Similarly, too little information sharing makes the ensemble ineffective, whereas too much information sharing results in information overflow or collaboration overload. Our experiences so far suggest that the concept of „sessions” offers a good point to balance activity synchronization and information sharing.

SCOPE is a prototype system of our ongoing research project and is currently being tested in a typical concurrent engineering setting, a software design process in our group. Most of our work to date has been focused on demonstrating the feasibility of implementing and using the system. We will further test and evaluate the usefulness of the system in more real-world settings. As a next step, we will develop more activity spaces as required by overall processes of software development and integrate SCOPE with the Smalltalk program developing environment. Another goal is to extend the use of SCOPE to other domains.

Acknowledgements

Many thanks to Weigang Wang, Christian Schuckmann, Daniel Tietze, and Martin Wessner for their valuable comments and support.

References