Modeling Cooperative Work Processes—A Multiple Perspectives Framework

Jürgen Ziegler
Fraunhofer IAO, Stuttgart

This article presents a framework, concepts, and notations for modeling cooperative work from different perspectives. The framework is based on the basic notions information, task, and actor, which are modeled individually and in relation to each other. The Cooperation Modeling Technique (CMT) built on these concepts provides notations for representing the different aspects of cooperative work. This results in a flexible approach for representing different analysis perspectives pertaining to the design of cooperation support systems. The method aims at providing abstractions and mechanisms that are targeted not only at structured aspects of work processes but particularly at unstructured and less formalizable cooperation issues.

The main focus of the approach is currently on asynchronous forms of cooperation, addressing informational awareness for asynchronous processes. The framework, however, also encompasses synchronous types of cooperation. The development of concrete methods for these aspects is a focus of future work.

1. INTRODUCTION

With the growing requirements for flexible, customer-oriented and knowledge-intensive business processes, work is becoming more and more cooperative in nature. The increasing complexity and low predictability of tasks in such contexts often requires tightly coupled, communicative and creative activities, involving people and roles from different parts of an organization or different organizations. These characteristics are particularly salient in knowledge-intensive work (see, e.g., Kidd, 1994) where processes are highly variable and require frequent adaptations. Many authors have pointed out that cooperative work in general is not following fixed, detailed procedures or processes and cannot be adequately captured and described by complete normative models (see, e.g., Gerson & Star, 1986; Suchman, 1987).

Many tasks performed cooperatively are only weakly structured or formalized and cannot be anticipated or modeled a priori, either completely or in part. In realistic settings, there is usually a combination of structured and unstructured activities.

Requests for reprints should be sent to Jürgen Ziegler, Nobelstrasse 12, 70569 Stuttgart, Germany. E-mail: juergen.ziegler@lao.fhg.de
that results in the need to use conventional applications such as a customer management system in conjunction with, for example, communication facilities or intranet-based document repositories. It is increasingly recognized that an analysis of cooperative work needs to take account of these two aspects in an integrated fashion (cf. Schmidt & Simone, 2000). It must be noted that the aspect of formalization or structuredness of cooperative work constitutes a continuous dimension rather than a dichotomy. Accordingly, systems for supporting cooperation within and across organizations can encompass a wide range of functions that go considerably beyond the functionality developed for conventional, domain-specific applications. The functionality provided may range from asynchronous communication through e-mail over structured and unstructured information repositories to synchronous telecooperation through videoconferencing and shared virtual environments.

The inherent complexity and openness of cooperative tasks does not imply, however, that formal analysis and design methods cannot be used or that designers do not need them. Just because of the wide range of functionalities and the variability of tasks, design methods are required that can guide the designer and serve as a means of communication between developers and the prospective users. However, current system development methods such as the Unified Modelling Language (UML; Booch, Rumbaugh, & Jacobson, 1999) or process modeling tools (cf. Scheer, 1999), providing generic modeling constructs and techniques, are mainly directed at the development of systems for well-structured tasks. They do not immediately provide concepts or techniques that are well suited to describing aspects of a domain or system that are pertinent to cooperative work. They are also not targeted at describing flexible and highly variable tasks or at the manifold communicative activities to be supported in a cooperative environment. Extensible modeling techniques such as UML provide the potential to introduce concepts and abstractions that are more directly matched to the characteristics of the domain of cooperative work. We claim that such extensions are needed and useful to provide designers of cooperation support systems with a structured approach that enables them to capture the manifold aspects of cooperation and coordination, and that incorporates means for expressing variability or vagueness where needed.

This article presents an approach to modeling cooperative work and designing cooperation support systems that provides a toolbox for analyzing and representing different aspects of cooperation in a flexible manner. The methods provide constructs suitable for undetermined or vague tasks, incremental refinement, different coordination mechanisms, and other pertinent characteristics of cooperative work. This work is based on the results of the project SPICE funded by the Work and Technology program of the German Ministry of Research.

2. REQUIREMENTS FOR ANALYZING AND MODELING COOPERATIVE WORK

With the increasing complexity and knowledge-intensity of today's work tasks, the possibilities of defining and modeling work processes based on such tasks a priori and in detail are decreased. Reasons for a low formalizability can lie in a large vari-
ability of the processes due to changes in the environment (e.g., customer requirements), complexity leading to an excessive number of possible execution paths, or because low-level details of processes are simply not known at the time of analysis. Figure 1 shows different aspects of work processes with either structured (or formalized) or unstructured properties. In many cooperative tasks, the unstructured aspects are predominant. Frequently, a precise description of detailed sequences will be dispensed with for cost-benefit reasons, especially when no specific system support is to be provided at this level.

In many cases, tasks or process components are executed cooperatively, that is, they include several persons or roles and are based largely on communication and coordination activities, whereas the complete process may follow some defined high-level flow. These cooperative activities are usually not known at design time; they emerge as the tasks are carried out. Examples of such tasks are, for instance, the joint authoring of a document or a group decision process that are coordinated by e-mail, phone calls, or personal meetings. Cooperative processes may comprise both workflow-like sequential or parallel task structures and open, unstructured tasks where task execution is organized on the fly using synchronous and asynchronous communication mechanisms. In many cases, the high-level steps of a process may be predefined in an organization whereas the details of accomplishing each major step are left to the group responsible for that step. In such cases, it is often the structure of the final or intermediate products of a cooperative task, which is specified beforehand (the "what"), rather than the concrete process to complete the task (the "how").

In general, different types of IT systems are required for supporting work tasks with different forms of communication. Although sequential dependence of tasks is often represented and controlled in workflow management systems, a multitude of technologies is available for cooperative task forms, ranging from simple communication and shared information resources to technologies for the support of group meetings (see, e.g., Borghoff & Schlichter, 1995; Ziegler, 1997b).

In the SPICE project, as well as in subsequent activities, a number of analyses of the requirements for supporting cooperation were carried out in six different com-

<table>
<thead>
<tr>
<th>structured</th>
<th>unstructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed, predefined tasks</td>
<td>variable, complex tasks, problem solving</td>
</tr>
<tr>
<td>clear responsibilities</td>
<td>ad-hoc teams, virtual teams</td>
</tr>
<tr>
<td>predefined workflows</td>
<td>spontaneous communication, document exchange</td>
</tr>
<tr>
<td>structured data</td>
<td>documents, Web content</td>
</tr>
<tr>
<td>specific applications</td>
<td>generic tools</td>
</tr>
<tr>
<td>tasks-specific knowledge</td>
<td>knowledge management, learning organisation</td>
</tr>
</tbody>
</table>

**FIGURE 1** Structured versus unstructured aspects of cooperative work.
panies from different branches. These analyses showed that due to the breadth, difference, and complexity of the activities observed, it would not be possible to model the relevant aspects of these work situations with a single method or even a limited set of techniques. From the analyses as well as from other sources, the following list of requirements was compiled that appear to be pertinent to modeling cooperative work processes:

- flexible use of different analysis and modeling perspectives according to the specific cooperation task and context.
- integration of structured, unstructured, and vague descriptions.
- abstraction mechanisms with the possibility of incremental refinement.
- level of detail of the models tailorable according to the stage and complexity of the design.
- reusable patterns for describing recurring cooperation structures.

3. A FRAMEWORK FOR MODELING COOPERATION SUPPORT SYSTEMS

This section presents a framework and methodology called Cooperation Modeling Technique (CMT) developed from these requirements for modeling cooperation-oriented processes consisting of different, integrated modeling views for cooperative activities. These views form different perspectives for the analysis of cooperative work as well as for the conceptual design of systems that support the cooperation.

The framework is based on three basic concepts: information (structured data, documents, and unstructured content); task (activities and processes); and actor (people, organizational units, roles). Each of these concepts or constructs can be analyzed separately from the others. This analysis yields models for each of the basic concepts, possibly using different facets of the concepts or different types of relations to form different models such as class or instance models. In addition, the concepts can be combined in binary relations, using again specific types of relations such information flows between actors or use relations between tasks and information objects. Ternary relations among these elements are also possible but are not further elaborated on here.

The basic elements can be arranged in a $3 \times 3$ matrix that allows for arbitrary combinations of these elements (Figure 2). The diagonal cells refer to structural models composed of one of the basic elements. The information-by-information cell contains models for structured and unstructured information, the task-by-task cell encompasses task types and task hierarchies, whereas the actor-by-actor cell describes types of actors such as roles or organizational entities and their concrete instantiation in the organization. The models in the diagonal each can contain a definition of different types (such as a class model for information or task types for the task cell), instances and their relations, and (potentially) a dynamic model, such as a state model for an information object.

In the other cells of the matrix, the respective models combine two of the basic elements. The element in the column header is also the main determinant for the structure of the model: in the information-by-actor cell, for instance, the informa-
FIGURE 2 The framework of the Cooperation Modeling Technique (CMT).

The framework structure is primary; the relations to persons or organizational units show whether an actor may access a certain piece of information or is notified about relevant changes of that information.

For each model outside the diagonal of the matrix there is a complementary model showing the secondary element of that model as its primary element. The model complementary to the one just described is the communication model, which has organizational actors as primary elements, different types of communication channels as relations, and information objects as flows.

Obviously, there is no unique way of selecting the semantics of the relations in these models. Hence, there can be more than one concrete model that fits into the framework cells. To be practically applicable, the framework has to be filled with concrete modeling techniques useful for designing cooperative applications. It provides, however, a space for analyzing cooperative work from a design perspective, as each of the models can be translated either into a technological support or organizational conventions and measures.

All framework cells have so far been filled with—at least initial—model definitions. For some aspects, well-defined modeling techniques have already been introduced and applied in concrete developments. Wherever feasible, existing standard methods are used such as the different modeling techniques of the UML.

In the following section, we describe sample methods from the framework in more detail. We focus here on three aspects of the analysis: the information model, a model of information logistics (relation between information objects and actors), and a cooperative process model called CoCharts.

4. INFORMATION RESOURCES IN CMT

Information models are one of the fundamental components of most analysis and design methodologies. Techniques such as UML offer a broad range of constructs to
represent object classes and their relations or instances of these classes. For structured information objects with a clearly defined set of attributes and operations, one can therefore apply the existing techniques without modification. CMT uses conventional UML class diagrams to represent such types of information resources.

In cooperative work, documents with different degrees of predefined structure are often one of the most important types of inputs and products of joint tasks. As shared artifacts, they also play an important role in coordinating the cooperative activities. It is therefore desirable to have techniques specifically suited to representing different types of documents. These types include standard sequential documents made up of components that may have different cardinalities such as optional components. On the other end of the spectrum, there are complex, nonsequential hypermedia documents that are composed of different media types and that may have different types of links between parts of the document.

Finally, there are often situations in the design of cooperation supports systems where one wants to abstract from the concrete types and instances of the information resources. Especially in the initial design of a system (such as a corporate intranet), the focus of analysis will be on the themes or topics to be represented in a system without specifying the concrete information resources provided for a particular topic. The technique for modeling topics in CMT is based on the topic map approach standardized in ISO/IEC 13250 (ISO, 2000)). Topic maps can represent arbitrary relations between topics and are therefore similar to semantic networks. A particular topic can be represented concretely ("occurs") in one or several information resources; in general an n:m mapping between topics and information objects can be developed. Topics constitute a metalevel description that can be added to the layer of concrete content without altering the structure of the original information objects. Topics maps deliver important input to designing thematic navigational structures.

Figure 3 shows the different information modeling concepts available in CMT. In addition to standard class diagrams from UML (upper left), topic map diagrams can be used for modeling the high-level conceptual structure of the information space (upper right). Due to the mapping between topics and information objects, each topic refers to a collection of concrete information resources. Topics can therefore be added to information models as placeholders for a collection of information resources.

Documents constitute important information structures in cooperative work. For this reason, there are two specific techniques for representing documents in CMT. One of these techniques describes the structure of composite documents by embedding the components inside the shape of the parent document. Multiplicities can be added to show how many of the document components can be present in an instance of the document type. These constructs are equivalent to a specific style of representing composites in standard UML. In addition, CMT information models can describe hypermedia documents with arbitrarily linked components. Currently, a technique for describing hypermedia documents using the constructs introduced is under development, building on concepts from techniques like ViewNets (Ziegler, 1997c) and OOHDM (Schwabe & Rossi, 1995). The details of this approach, however, are beyond the scope of this article.
4.1. Information Modeling as a Basis for Awareness Requirements

A model of the information managed in a cooperative application is an important prerequisite for determining informational awareness mechanisms (especially asynchronous mechanisms) such as notification techniques. By means of the underlying information model, one can describe more precisely which type of information objects or topics are used in an awareness-supporting application.

Topic maps can be used as a basis for developing metadata structures, which are one important means of creating user interest profiles for supporting theme-oriented awareness. Subscription and notification techniques are important approaches for keeping the user up-to-date about changes or new entries of documents in their area of interest. Implementations of topic map-based awareness mechanisms can be more powerful than, for example, keyword-based profiles of interest as the semantic relations in topic maps allow the use of inferencing mechanisms for not only delivering the documents directly related to one topic or keyword, but also for deriving larger sets of information objects from subtopics and other semantically related topics.

5. MODELING INFORMATION LOGISTICS AND INFORMATIONAL AWARENESS

In many current applications, information is not just a passive resource that can be created, modified, or read by the users but can play an active role by notifying users with a particular profile of interest through a variety of mechanisms. It is
therefore a useful notion to think of the interaction between information objects and organizational roles as a two-way communication. The relations between actors/roles and information objects comprise standard read/write mechanisms, including authorizations as well as active functions of the information. The main active functions of the information resources are push techniques and notifications. Push techniques are assumed to send the complete information or parts thereof automatically to the recipient based on some triggering event or condition. Notification mechanisms can take various forms, the most common being sending some reference to the information object (which can usually be activated by the user through some simple mechanism like clicking a link). Other forms of notification may use more descriptive pointers to the original source like summaries generated by a human author or some automated summarization technique. In fact, rather than being two distinct mechanisms, push and notification form a spectrum of techniques that result in different degrees of the user’s awareness of the actual information content and the processes related to it. Determining appropriate levels of awareness is one of the most pertinent issues in designing cooperative systems, as a too-high level of exposure to the details of cooperative processes may overload the user and low levels of awareness may negatively affect coordination, efficiency, and other aspects of cooperative work (Dourish & Bellotti, 1992).

Figure 4 shows a model of the information logistics (information-by-actor) for an information resource “market info.” Actors can interact with the information resource by standard read/write access or by relations representing notification or moderation. Notification mechanisms can be further specified through trigger conditions or by indicators whether the full information or just a reference is pushed to the receiving organizational unit. This model is useful for standard information access but particularly for areas like intranet-based knowledge management, which needs to take into account different information management and awareness mechanisms.

6. MODELING COOPERATIVE PROCESSES

6.1 Characteristics of Cooperative Processes

The most diverse methods and techniques for the modeling of work processes, business processes, or both are available and can be used for analyzing and designing processes as well as for the selection and design of respective support systems. Often one aim is to control processes with the aid of workflow management systems. Familiar representatives of modeling techniques for work processes, business processes, or both are event-driven process chains (Scheer, 1999) or Petri net-based approaches (e.g., Gruhn, 1993). Overviews on modeling techniques for business processes are, for example, included in Jablonski, Böhm, & Schulze (1997).

Cooperation-oriented processes have a number of specific characteristics in which existing modeling techniques show distinct deficiencies:
Cooperative Work Processes

![Diagram showing relations between actors and information objects.](image)

**FIGURE 4** Relations between actors and information objects. The set of relations defined allows modeling aspects of information logistics such as access rights or notification services. The example on the right-hand side shows the relation of several organizational units with a marketing-related information repository (Unified Modelling Language [UML] package of information resources).

- Cooperation-oriented processes often form a mixture of pre-arrangeable steps to be worked off as a workflow and open, mainly team-oriented tasks that are coordinated by a variety of different mechanisms. For this, descriptions are needed that are able to represent workflow-like sequences as well as cooperative task execution in integrated form.

- To be able to represent and survey different levels, abstraction mechanisms are necessary for the tasks as well as for the information objects used, events, or processing conditions.

- There must be the possibility for the incremental formation of models and it must allow for vague descriptions (e.g., stating subtasks without precise description of their sequence; cf. also Herrmann & Loser, 1999).

Due to the complexity and variability of cooperative processes, the economy of developing a model has to be improved, for example, by reuse of recurring patterns or abstraction from low-level details.

### 6.2 Task Modeling in CoCharts

The approach presented here is based on the task modeling technique of task-object charts that are introduced in Ziegler (1997a). Task-object charts use the higraph formalism developed by Harel (1988), which is also used in the widespread state mod-
eling technique of statecharts (Harel, 1987). Task-object charts are based on the following concepts also applied in CoCharts (Figure 5).

The main elements in CoCharts are the tasks as parts of processes, which can be annotated by an attribute indicating the organizational unit or role in charge of the task (Figure 5a). Complex tasks can be refined in separate diagrams (Figure 5a, lower part).

According to the statechart notation, subtasks can also be embedded directly graphically in the supertask, thus enhancing clarity and coherence considerably. Two cases must be differentiated. Figure 5b shows the decomposition into sequential subtasks, that is, at any given time only one of the subtasks can be active. It must be noted that this may not yet define the sequence of the tasks if the transitions showing the flow of control are omitted. All subtasks can be worked off in one sequence, or only individual subtasks can be activated based on conditions. The OR decomposition thus describes a synchronism constraint (tasks must not be executed simultaneously; regarding constraints in workflow; see, e.g., Dourish, Holmes, MacLean, Marqvardsen, & Zbyslaw, 1996).

The second type of decomposition of a task is the concurrent decomposition (AND-decomposition, Figure 5d), in which all subtasks are activated simultaneously, that is, can be processed in parallel (if the supertask is activated). Parallel processing occurs, for example, when different components of an object can be processed independently of each other (e.g., different parts of a large document). Different aspects of a shared object can also be processed in parallel, if necessary (e.g., editing of a document with respect to form or content), or a time-interleaved or overlapping processing (e.g., in a group meeting) can take place. In all cases of parallel tasks, actions for the synchronization and merging of the partial

![Figure 5](image-url) Notation of tasks and task decompositions in CoCharts.
results are necessary, which can range from simple aggregation to complex consolidation activities (e.g., in a group editor).

Finally, Figure 5c illustrates that the type of task decomposition remains unspecified for the time being and only the subtasks are known. This can be used for incremental specification with subsequent specification or for deliberately uncertain models. In both types of task decomposition, additional optional elements can be specified.

6.3 Flow Relations

Relations between simple or hierarchic tasks are interpreted as flows (this is presented here merely in an abbreviated manner, a detailed discussion can be found in Ziegler, 1997a). There are different types of flows that can be used for modeling the control of task sequences:

- Control flows representing general trigger events that can start the execution of a task (notation “event!”). These can be temporal events (“end of month!”) or communicative events (“info request!”) generated by an actor participating in a cooperation.
- Flow of information objects between tasks. This type of relation indicates that the successor task can start as soon as an object of the type specified is made available by its predecessor.
- Flow of objects with states: This type of relation indicates that the successor task can start as soon as an object that is in the specified state of processing becomes available. A definition of this state is done separately, for example, in a state diagram for that object class so as not to overload the process model with details.

Object states have here the function of pre- and post-conditions for the execution of a task: If there is an incoming flow of object class O in state Z for a task, this means that the execution of the task requires the availability of an instance of the class O in state Z. Executing the task transfers the object to a new state (post-condition of the task), which in turn can trigger subsequent tasks. Depending on the state generated, different subsequent actions can be initiated (Figure 6).

In hierarchically structured tasks, input flows can either end at the border of the supertask or at a specific subtask. In Figure 6, an incoming object instance of type “insurance application” triggers a superordinate task “check application,” which contains concurrent subtasks. Both subtasks are activated as soon as the supertask is started. Depending on the state of the application produced (“App_OK” or “App_NOK”), different successor tasks are activated. If more than one state can be

1 A similar notion is used in the activity diagrams in UML in which each task in a process can be associated through a use-relation stereotype to an object. These objects can also be further specified by annotating them with states. However, the concept of object states triggering the flow of control in CoCharts is not explicitly used in UML. Also, using object states as direct flows between tasks often makes the visual representation more transparent and compact. The UML notation is used in CoCharts only if further details of the object are to be represented or if the (active) object triggers some task execution.
produced by a task, they are by default disjunctive; that is, exactly one of the result states can be reached. Depending on the type of decomposition of the task, the logic of the outgoing flows can be different (this is not further discussed here for reasons of space).

The relation between the task "prepare contract" and the object class "contract" shows that a new instance of that class that is in the (initial) state "offered" is produced. The plus sign at the connection arrow is a stereotype that indicates the creation of a new instance.

The notation of classes, instances, and states follows the conventions of UML: Underlined object names denote instances. \texttt{c:contract}, for example, denotes a (prototypical) instance "c" of class "contract." According to the UML concepts, anonymous instances such as \texttt{:contract} can be specified. For further details on naming conventions for classes, instances, and related concepts, we refer the reader to the UML documentation.

With the concepts introduced in CoCharts, powerful abstract mechanisms are provided that enable the presentation of processes on different aggregation levels and with different degrees of specification. These means make it possible to comply with important requirements for modeling cooperative processes by, for example, defining only the necessary tasks of a part of the process but delegating planning or execution of the actual processing to the group. Particularly, it is possible to define only a target state as a post-condition of a task and to leave the definition of the subtasks to the cooperation partners. The following paragraph shows how high-level information for the execution of cooperative tasks can be provided by introducing coordination patterns that don't require specifying the execution in detail.

7. **COORDINATION PATTERNS AND AWARENESS REQUIREMENTS**

The cooperative execution of work tasks requires coordination activities that ensure that dependencies resulting from, for example, technical, temporal, or organi-
zational conditions are taken into consideration (Malone & Crowston, 1994). Various authors have attempted to identify and classify recurring basic coordinative processes (Crowston, 1994; Schmidt & Simone, 1996). Considering the multitude of analyses and various approaches regarding the description of coordination and dependencies in organizational science and related areas, which are of special importance to the design of cooperation-supporting information systems, those aspects are focused on in the following.

The following describes an approach of how CoCharts can be expanded by the use of coordination patterns. A coordination pattern defines the structure and behavior of a basic, recurring coordination process. Analogies can be found, for example, in the programming field in form of design patterns (Gamma, Helm, Johnson, & Vlissides, 1995) describing abstract, reusable object structures and constellations. Here, coordination patterns serve the purpose of achieving condensed, abbreviated notations for cooperative task processing. They are employed as a means of abstraction so that only the type of coordination has to be stated in cooperative tasks, without having to describe the processes individually. This is useful for two reasons: In many cases, the actual coordination processes are too complex and variable to be described fully. In other cases, a convenient “shorthand” is needed for constantly recurring process components (as, e.g., standardized recording procedures). Mechanisms and tools for the execution of coordination patterns are described separately and can, therefore, be multiply used.

7.1 Notation of Cooperation Patterns

In CoCharts, the actual sequence of cooperative processes can be specified gradually with increasing precision. As an additional modeling component, complex tasks can be described by specifying a coordination pattern that determines in which way subtasks connected with the complex task are to be coordinated. If no subtasks are specified, these have to be determined in the cooperation; thus, the determination of the task is part of the coordination process. If subtasks are known, coordination is limited to the regulation of the processing.

In addition to specifying a pattern as a mechanism of coordination, one or several roles can be defined that take over the coordination function for this pattern. In case of a centrally coordinated task, this can be a project manager, for example, or, for heavily interleaved group tasks, the team itself. The specification of a name of a pattern refers to the respective entry in a catalog of coordination patterns that include the pertinent detail descriptions.

7.2 Approach to a Classification of Coordination Patterns

To use coordination patterns in the description of cooperative processes, a classification and catalog of recurring patterns needs to be developed. This section describes an initial attempt to provide the basic concepts and dimensions along which such a classification can be achieved. The set of patterns presented here,
however, is not necessary complete or optimal. The elaboration and evaluation of such patterns is subject to further research.

**Coordination through process and product structure.** One of the most pertinent distinctions in cooperative work is the notion that processes are not only constrained by a predefined temporal structure of tasks but also by the structure of the (information) product the process is intended to achieve. In many cooperative situations, such as jointly writing a research proposal, only the structure of the product may be predefined, whereas the process to achieve the result is left to the group working on it. In many contexts, even the structure of the outcome is not known at the outset, but is developed “on the fly.” This is typically the case, for instance, when a team of authors writes a journal article. In many instances, the modeling of a cooperative task can be limited to specifying the structure and other constraints of the end product, rather than defining an explicit process. The coordination needed to produce the result intended is then achieved through standard mechanisms such as a joint repository for the intermediate and final result and open, unconstrained communication mechanisms using standard tools such as telephone or e-mail.

With respect to coordination mechanisms, the structures of the process and the product play a symmetrical role; both can constrain the activities performed by a group to achieve an intended outcome. However, when only the product structure is known from the beginning, the resulting type of cooperation is typically much more flexible and communication oriented (both asynchronous, e.g., through e-mail, and synchronous, e.g., in joint meetings).

Both types of coordination constraints typically occur together in realistic cooperative processes. Often a high-level process is defined containing only the major steps in that process, whereas the products produced by these steps may be specified by their structure. Let’s consider the example of a development project in a company: A high-level procedure may be defined comprising tasks like writing a project proposal, reviewing the proposal, accepting or rejecting it, doing the development work, reviewing the results, and so on. Lower level aspects of the process may be defined by prescribed structures of the project proposal, the project report, the review document, and so on. Often, a given structure of the outcome of the process is associated with organizational responsibilities for the different parts of the product.

Both process- and product-based types of definitions typically appear in the same cooperative process. In the first case, the coordination is achieved through specified communication actions (such as routing a document to the next role involved in the process). In the second case, the group needs joint access to the definition of the product and the various intermediate stages of the evolving product. Both aspects have characteristic awareness implications: Process awareness indicates to what extent the cooperation partners can observe or are informed about the state of the process. Product awareness requires techniques to achieve an appropriate degree of transparency concerning the current content of the different product
parts or concerning metalevel data such as state attributes associated with the product parts.

**Dimensions of coordinative activities.** The *coordination type* differentiates cooperation based on explicit communication actions (e.g., forward a document) and cooperation controlled by the access to a shared information object. In the first case, cooperation partners "share" the tasks of a process; the individual tasks are prestructured and allocated to different people, or the execution of a task is handled by different people in an interleaved manner. This case is called "shared process." In the second case ("shared object"), cooperation is controlled through the status of a common object. In prestructuring the object into independent components, the case of a parallel, decoupled work is reached, whereas in dependences between the components, more interleaved process forms—up to synchronous group editing—are needed. Access and processing rights can ensure the assignment of object components to the individuals.

The *coordination time* specifies at what point of time the regulation of the cooperation sequence occurs. This can occur for an entire class of processes (this is the rule for the definition of set workflows), before the execution of the respective component of a process instance ("deferred"), or can ensue during the execution of the tasks from the behavior of the cooperation partner ("emergent").

The *coordination agent* specifies the organizational units or roles responsible for coordination. It determines the centrality versus decentrality of the coordination process. Typical characteristics are coordination by individuals ("process owner"), coordination by organizational hierarchy ("hierarchy"), or distribution by the executive team (distributed).

Along these dimensions, a large number of different coordination patterns can be aligned. A few samples are:

- An ad-hoc workflow for a complex subtask is defined by a supervisor (process owner) prior to executing it (type: shared process; time: deferred; carrier: owner).
- In one step of a process, the structure of a document to be authored jointly is developed in a group meeting (type: shared object; time: emergent; agent: distributed).
- Planning has to be authorized by superiors on several levels before subsequent activities can be started (type: shared process; time: planned; carrier: hierarchy).

Using the dimensions introduced, patterns of cooperation can be defined that can be used as building blocks for designing cooperative processes. Separate CoChart representations can be made up to define these patterns. The patterns introduced here are just examples illustrating the concept. Each pattern is typically associated with a particular set of tools to support that type of cooperation by means of information technology (such as groupware systems). Further work will have to
provide a more comprehensive collection and precise definition of frequent or typical patterns and the associated technological tools.

### 7.3 Example of Using Coordination Patterns

Figure 7 shows, as a sample for the application of CoCharts with coordination patterns, a (simplified) extract of a product development process of a company. The partial process starts with marketing issuing a request for a new product development.

The request is then dealt with in a decision task, which uses a “notification and meeting” pattern. The request is forwarded to the actors involved in the decision, in this case the development team. Once a positive decision has been made, a new development project is started that is associated with a development document that has a predefined structure for the type of product to be developed. Assuming no further procedural constraints, the task “develop product” is performed by a unit “development team” according to a pattern called “project database and direct communication.” In this pattern, which needs to be defined separately, all members of the team have read-and-write access to a database containing the development documentation. The document has three fixed parts.

Assuming that there are complex dependencies among the three different design aspects of that product, the concrete coordination of how the documentation is developed is performed by direct communication, that is, the team responsible communicates in an unconstrained way by whatever tools are available for this purpose.

In addition to specifying the structure of the outcome of this task, the CoChart notation can indicate additional information needed for performing the task, either by specifying concrete data or documents associated with the type of task, or by indicating topics that are pertinent for the tasks. According to the information resource model for the cooperative process (not shown here), these topics point to a

---

**FIGURE 7** Example of a CoChart process model using coordination patterns.
number of concrete documents that must be accessible to the development team. Suitable access and navigation mechanisms can be designed that make this access efficient and user friendly.

Concerning the access of the development team to the jointly available documentation, different levels and mechanisms for creating group awareness concerning the progress of the cooperative task can be devised. Typical awareness mechanisms include, for example,

- simple read access to the parts developed by the other team members.
- explicit status flags associated with the different parts of the document indicating progress and possibly the need that some member of the group start a new subtask.
- review cycles initiated by the author of a document part, routing that text to the other members of the group (possibly with specified response deadlines).
- a jointly accessible project plan that details the subtasks, their timing, and the persons in charge of the different parts.

Further work is needed to systematically classify these mechanisms and to introduce them in the model as a means of specifying more precisely the nature of the coordination pattern. Having a well-structured catalog of pattern with well-understood awareness characteristics will also support the design of such processes in such a way that appropriate levels of awareness are provided. In this way, sufficient information can be provided to manage the dependencies between tasks and to avoid information overload.

With the chosen representation, a good overview of the process is available without unnecessary details obstructing the view of important steps. The process shown here is a typical example for a transaction with a mixture of workflow-like segments and cooperative, undetermined tasks.

8. DISCUSSION

This article presents a framework, concepts, and notations for modeling cooperative work processes. The framework is based on the basic notions of information, task, and actor, which are modeled individually and in relation to each other. The CMT built on these concepts provides notations for representing the different aspects of cooperative work. This results in a flexible approach for representing different analysis perspectives pertaining to the design of suitable cooperation support systems. The method aims at providing abstractions and mechanisms that are targeted not only at structured aspects of work processes but particularly at unstructured and less formalizable cooperation issues.

CMT builds on existing standard modeling techniques where appropriate, but extends them to also capture those aspects of cooperation contexts that are less formalized or unstructured at the beginning. Standard UML class models are extended and complemented by document-based information types and a metalevel description for analyzing the topic structure of a domain. A model of information
logistics shows communication relations between actors and information resources, capturing important aspects of informational awareness.

The technique CoCharts as part of the method offers the possibility of representing work processes with a mixed form of workflow-like components and cooperative, not completely determinable, tasks. In such processes, typically different forms of coordination occur, ranging from fixed, sequential processes to interaction in team meetings. The mixed occurrence of these different coordination forms is typical for today's organization principles and will gain importance in the future with the increase in knowledge-intensive work processes. By developing a catalog of coordination patterns, a common basis for modeling can be created and the economy of building a representation can be enhanced.

The framework encompasses further analysis aspects, especially for representing organizational roles and their structure, tasks allocated to these roles, and information resources needed for performing the range of tasks associated with a role. By detailing this aspect of the analysis, learning and knowledge exchange requirements also can be defined in a consistent manner. The specific techniques for representing this aspect cannot be shown here for reasons of space and will be reported elsewhere.

The main focus of the approach is currently on asynchronous forms of cooperation, addressing informational awareness for asynchronous processes. The framework, however, encompasses synchronous types of cooperation also. The development of concrete methods for these aspects is a focus of future work.

REFERENCES


