# Flexible Coordination with Cooperative Hypermedia

*Weigang Wang* 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: {wwang, haake}@darmstadt.gmd.de

**ABSTRACT**
In current workflow and groupware systems, there is a gap between formal and informal coordination mechanisms. To fill the gap, flexible coordination support covers the whole spectrum of informal and formal coordination mechanisms. In this paper, a flexible coordination model integrating formal and informal coordination mechanisms is presented. Methods of using cooperative hypermedia concepts to uniformly model all objects representing coordination mediums and shared artifacts are described. Using the proposed model and methods, a cooperative hypermedia system (CHIPS), that offers flexible coordination support has been implemented. An application example of the system shows how a set of tasks and different coordination mechanisms are integrated into a cooperative process. This work demonstrates that cooperative hypermedia can serve as a bridge to close the gap.

**Keywords:** Cooperative hypermedia, groupware, coordination, workflow, CHIPS

## INTRODUCTION

Recently, there is a growing interest in new organizational forms, such as virtual organizations or virtual companies. In contrast to traditional organizations that place their emphasis on *tasks* to be performed by *individuals* in relatively static hierarchical organization structures, the new organizational forms place their emphasis on efficient *processes* performed by *teams* dynamically formed for specific projects [15]. Such processes rely on efficient coordination.

### Motivation and Problem Statement

There is a wide range of coordination mechanisms, from very informal to very formal, that are suitable for different work situations. Initially, groupware systems, including many cooperative hypermedia systems, tried to improve group interaction via informal coordination support, such as bulletin boards, calendar tools, and shared documents on an ad-hoc basis. As the scope of groupware applications spreads towards more formal business focused group interactions, there has been an increasing requirement to provide a more formal and controllable procedural framework to support the use of groupware applications [9]. Workflow systems, on the other hand, were initially targeted to support more structured work via formal coordination support, i.e., through computer controlled execution of prescribed processes. As the scope of workflow applications extends to cover less structured work situations and as work situations are becoming more dynamic, there has also been an increasing need for supporting more flexible process and less formal group interaction [14].

In practice, a task (or process) often includes a mix of un-structured, semi-structured, and structured parts. For more structured parts formal coordination mechanisms are useful while for less structured parts informal coordination means are required. New projects or tasks often lead to new subtask or process structure that is not completely known in advance. Furthermore, existing process structure may change over time. Thus, flexible coordination is required to deal with a changing degree of structuredness (from more to less structure and vice versa) and with changes in the process structure itself (i.e., emerging processes). To meet the needs of various work situations and to tackle work situation dynamics, different coordination mechanisms should be provided and coordination support should be flexible. Although groupware and workflow systems have provided some coordination mechanisms, there is a gap between informal and formal coordination mechanisms that are currently supported.

### Our Approach

Our solution to solve the problem is to model flexible coordination with cooperative hypermedia. Our approach can be outlined as follows:

- By examining coordination theories, models, and means, a flexible coordination model that integrates formal and

informal coordination mechanisms is derived.

- Cooperative hypermedia concepts are used to uniformly model all objects that represent coordination mediums and shared artifacts.
- Informal coordination is supported by shared hypermedia workspace, and other cooperation features, such as access control, concurrency control, group awareness, notifications, and mediator agents.
- The capability to enact processes is supported by attaching computational semantics to the hypermedia objects representing the process structure.
- Because of the unified representation of coordination medium and shared hypermedia structure, the combination of and transformation between informal and formal coordination mechanisms are made easier to handle.

In order to test our approach we built the CHIPS prototype employing our COAST toolkit, which supports the implementation of asynchronous and synchronous groupware [17].

The rest of the paper is organized as follows: Section 2 discusses related work in the areas of hypermedia, groupware and workflow. Section 3 presents existing models and means for coordination, and proposes an integrated organizational coordination model that combines both informal and formal coordination means. Section 4 describes how to use cooperative hypermedia to realize the integrated model. In Section 5, the CHIPS system and its flexible coordination mechanisms are described and an example is given that shows how a set of tasks and different coordination mechanisms are integrated into a well-coordinated cooperative process. Section 6 ends the paper with a comparison to related work, a conclusion, and some plans for future work.

## RELATED WORK

Coordination support is the primary area of workflow management systems. Workflow systems can support more formal coordination but lack support for collaboration and informal coordination. When describing the future work of workflow systems, Sheth points out that one direction is to have integral support for collaboration, not just coordination [18]. Nutt argues that workflow systems should be made more flexible, and the goal of leading edge systems is to support existing social models in organizations through informal communication, to have high computation support for specific tasks, and to provide variable coordination support. Such systems should be able to support both situated work (which can not be truly represented in a prescriptive workflow) and workflow (or some customized combination of the two) on a case-by-case basis [14]. Ellis argues that successful workflow systems must extend the process models to include declarational specifications about the goal of a task (or a process) alongside operational ones [3].

Cooperative hypermedia systems and many other groupware systems can provide some informal coordination support, but in general lack support for more formal coordination. Much research has been done in cooperative hypermedia systems with respect to supporting concurrent access and manipulation of hypermedia documents. Among this work, we find support for asynchronous cooperation [19, 1, 23, 5, 25, 8] and synchronous cooperation [21, 22, 12]. The coordination means provided by cooperative hypermedia systems are a collection of low level technical mechanisms, such as history cards and access locks in Notecards [23], access control and concurrency control in ABC [21], and group awareness, cooperation modes, and shared information spaces (e.g. for planning) in SEPIA [22]. However, these systems focus on ensuring consistency and neglect coordination of and within work processes. This is usually handled by human users on an ad-hoc basis. Other systems like Anecdote [8] focus on support for designing, specifying and planning the document structure and content. While Anecdote allows simulations and supports checking the status of a project by comparing planned against existing document units, it does not provide means for synchronous work or further coordination and collaboration support.

In order to bridge the gap between formal and informal coordination mechanisms and to provide integrated communication, coordination, and cooperation support, recently there have been many research and development efforts on merging workflow and groupware [3]. The basic approach taken by the workflow community is to integrate groupware features into existing workflow systems for informal coordination or exception handling, such as the recent work in the Exotica project to integrate FlowMark and Notes [18]. Our approach to the integration is just in the opposite direction. We start with a cooperative hypermedia system offering many communication and cooperation features, and integrate process support capabilities (i.e., more formal coordination support) into the hypermedia system. This approach leads to a system dealing with processes and information structures (as the subject of processes) as two views on a unified data model, permitting smooth transitions between these two views, and supporting cooperative work on defining and manipulating process and information structures. Unlike an extended workflow system, which still maintains the strict separation between process and document data, such a system can provide much more flexibility.

Trellis used a Petri-net-based model to represent document structures with browsing semantics, cooperation protocols, and software processes [4]. On modeling formal coordination, there is some similarity between Trellis and CHIPS, while Trellis has not particularly focused

on supporting simple manipulation of evolving process structure by end users, smooth transitions between different cooperation styles and simple integration of evolving process structure and document space.

## FLEXIBLE COORDINATION FRAMEWORK

In order to have a solid theoretical basis for flexible coordination, first, existing coordination theories, models and means are examined and then an integrated coordination model is developed which has components for modeling both informal and formal coordination mechanisms.

### Coordination Theory and Coordination Means.

Coordination is the act of working together harmoniously and coordination theory is the body of principles describing how this should be achieved with respect to how dependencies among activities can be managed [11]. There must be some actors, performing some activities which are directed towards some ends or goals. If there are no dependencies, there is nothing to coordinate. Malone and Crowston has identified three basic types of dependencies among activities: flow, fit, and sharing [11]. *Flow dependencies* arise whenever one activity produces a resource that is used by another activity. *Fit dependencies* arise when multiple activities collectively produce a single resource. *Sharing dependencies* occur whenever multiple activities all use the same resource. Different dependencies can be managed by different coordination mechanisms [11]. Flow and fit dependencies are the focus of most existing process mapping techniques. Sharing dependencies can be managed by a variety of mechanisms such as concurrency control and access control mechanisms.

Coordination is a complex process which consists of a number of subprocesses such as identifying goals, ordering activities, assigning activities to actors, allocating resources, and synchronizing activities. Coordination processes require that some decision has been made and accepted by a group. Group decisions, in turn, require members of the group to communicate in some form [10]. Two typical computer-mediated communication forms are 1) direct communication between actors; and 2) indirect communication with information that can be observed by every actor. These communication means serve not only the needs for exchanging information, but also the needs for coordinating activities.

There are two extreme types of tasks: unstructured tasks and structured tasks. Structured tasks, such as that for approving business trips, often have a detailed model that clearly describes the steps that are necessary to complete them. Unstructured tasks, such as writing a conference paper with different colleagues, have no obvious structure, and they are never done in the same fashion [16]. A suitable coordination mechanism for structured tasks is formal coordination, in which coordination is handled by actions initiated according to explicitly defined dependencies. A suitable mechanism for unstructured tasks is informal coordination, in which there is no predefined flow of work and coordination is handled by actions initiated by people themselves on an ad-hoc basis. Informal coordination is supported mainly by computer-mediated communication systems (or groupware); while formal coordination is supported mainly by workflow systems. Between the two extreme types of tasks there is a wide range of semi-structured tasks. A task often has some structured parts and some less structured parts (subtasks), and the structure of a task may also change over time along the informal-formal dimension. It is between the two extremes that lies a gap for flexible coordination support. This leads us to consider three kinds of technology for coordination.

*Informal Coordination through Computer-mediated Communication.* Informal coordination with computer-mediated communication is largely handled by people themselves. Basic means for direct communication between actors are audio/video links for synchronous communication and email for asynchronous communication. Basic means for indirect communication include shared artifacts, various types of group awareness, and notifications.

The access to shared artifacts can be coordinated with either technical protocols based on access locks and transactions or social protocols supported by group awareness. The use of shared artifacts can also be coordinated in many different cooperation modes ranging from de-coupled, to loosely-coupled, and to tightly-coupled modes. In a loosely-coupled mode, one might prefer to coordinate manipulation of shared artifacts with technical protocols, while in a tightly-coupled mode, it might be preferred to switch to social protocols [16]. Notifications are messages that are generated automatically through user interaction with the shared artifacts.

*Formal Coordination using Workflow Technology.* A work procedure in a workflow system is defined by a workflow model composed of a set of discrete work steps with explicit specifications of how a unit of work flows among the different steps [2]. In general, a workflow coordination model can be defined as a directed graph, (N, L), with a node set N representing individual steps in the procedure and an edge set L representing the coordination structure among the tasks [14]. One of such models is a Petri net (or its logical equivalent), which is a control flow model that explicitly represents sequence, AND- or OR-flow. Graph models provide a formal basis for defining control flow; they also provide a set of graphical representations for workflows.

The strength of workflow systems lies in their support for automated execution, control, and communication of

work as is required to satisfy workflow processes. The limitations of many existing workflow systems remain in their lack of support for more dynamic environments and their lack of support for human involvement in organizational activities [18].

*Coordinated Organizational Communication.* Coordination and communication often happen among actors and activities in an organizational context. Therefore, the elements and structure of organizations should also be considered. Two notable approaches to model such coordinated computer-mediated organizational communication are: 1) mediator-oriented approach, such as in AMIGO [20] and 2) role-oriented approach, such as in AME [20]. In the first approach, regulation knowledge in organizations is represented by an event script which an autonomous agent uses to direct communication, while the main concern of the second approach is to vest responsibility for the performance of an activity in the roles involved in taking part in it (i.e., so that people taking the job (role) know what to do and with whom to communicate).

The advantages of the first approach lie in the fact that it is simpler to manage the synchronization and coordination conditions and to recognize exceptions, errors, and deadlocks, but the major disadvantage lies in the lack of individual support for human actors playing roles in the activity. The activity can progress only when the individuals do what they are requested to do. In the second approach the communication among the group members is managed by the roles themselves. The power of this approach is that the role specific splitting of work is reflected in the regulations of each role. The disadvantage is that detecting error situations and deadlocks, and controlling or tracing could be rather difficult because the knowledge about the activity and its state are completely distributed among the various roles.

### Our Approach: Integrated Coordination Model

In order to support flexible coordination, a model that integrates the above three categories of coordination means is proposed. This model is built mainly on a role-oriented approach, however, a specific mediator role is added to handle high-level centralized issues. This mediator role can be played by both agents and people. In the model, direct and indirect communication means are incorporated as tools provided in the workspace. The model encompasses both the structural elements of an organization, and activities that these elements may be involved in. This provides a basis for better group awareness across organizations. The role-based approach is also augmented with a workflow-based approach, such that the dynamic high-level coordination and communication between people for less structured tasks can mainly be handled by people themselves; while the formal coordination for more structured tasks are handled mainly with a workflow-based approach. Queries to global information are handled by the mediator agent. In the model, the following components are identified:

- *People*: placeholders that represent actual individuals working within one or more organization unit(s), and their capabilities,
- *Agents*: software that operates on behalf of people on designated tasks (activities),
- *Roles*: specifications of the responsibilities assumed by any person or agent that plays the role,
- *Organizations*: organizational units, groups or project teams together with their organizational structures and staff relationships,
- *Activities (tasks)*: tasks or steps of tasks that are performed by actors,
- *Processes*: a set of activities which collectively realize a goal, normally within the context of an organizational structure defining functional roles and relationships,
- *Actors*: a collection of individuals (or agents) who are assigned to perform a task.
- *Shared Information Space*: it provides an information repository and basic data manipulation mechanisms. It contains documents, reusable process schemata and templates for performing various tasks. An *Organization manual*, which contains information on organization structures, responsibility of roles, projects, resources, etc is also maintained in the shared information space.
- *Workspace*: virtual space or conceptual work area where all the activities take place. It is a part of the shared information space and contains private or shared documents and tools. Furthermore, by incorporating task and process models in it, a workspace becomes an
  - *Activity space*: which is a workspace with task-specific knowledge structure and functions, and/or a
  - *Process space*: which is an activity space that can represent and execute processes.

The basis of the model is the shared information space. Built on the shared information space, workflow in a process does not necessarily mean the real transport of all task-related information objects. For instance, some information objects can stay where they are, while their addresses are forwarded to the actors of following tasks, or their access permission is changed according to the next actors. Activities and processes can be performed in activity spaces (or process spaces) in different cooperation modes. Coordination now builds on the responsibility of acting roles, predefined communication protocols, and process definitions. Roles are distinguished from people, thus allowing a separation of responsibilities between a role and a person playing it. The mediator is crucial for coordination. People can request information from the mediator, such as who is currently logged into the system, what role they are playing, and what part of the information space they are accessing.

The mediator also handles the high level concurrency control of the environment. When conflicts arise, they will be detected by the mediator, related parties will be notified and social mediation can be initiated.

## COORDINATION WITH COOPERATIVE HYPERMEDIA

This section describes how cooperative hypermedia is used to represent the components identified in above integrated coordination model. In our approach, we start from a basic typed hypermedia data model including basic computational capabilities (e.g., simple manipulation operations). Based on cooperation functionality, applications using the basic data model can be shared by groups. Task-specific extensions of the basic data model lead us then to activity spaces offering task-specific document models and computations. Employing the cooperation functionality, activity spaces become shared workspaces. Finally, we define a specific activity space for defining and enacting processes. This process space provides a process model that represents processes as typed hypermedia structures with process computational semantics. It also facilitates the integration of processes and the documents they operate on since both are modelled as hypermedia structures. Furthermore, different types of processes can be integrated within an overall process structure. As our hypermedia model is highly in conformance with the Dexter model [7], the methods described here should be applicable to many Dexter-based hypermedia systems.

### Shared Information Space: Cooperation Framework

The cooperation functionality employed in our approach is based on the COAST framework for synchronous and asynchronous groupware [17]. In general, two layers can be distinguished in this framework (from bottom to top): a basic CSCW support layer and a hypermedia layer. In the basic CSCW support layer means for representing shared artifacts, for maintaining consistent replicas and for accessing shared object spaces via session objects are offered. Session objects register the participating users of a session (and their corresponding browsers, see below), define the coupling of the participating browsers and the degree of group awareness offered, and they provide designers with a high level interface to handle access control and concurrency control. In addition, graphical users interfaces (session browsers) are provided for users to access and manipulate shared object spaces. Using these services, the hypermedia layer provides shared hypermedia object spaces (e.g., a typed hypermedia document model offering e.g. nodes, links and composites) and the corresponding cooperative hypermedia browsers and types of session objects supporting cooperative navigation and manipulation of shared hypermedia documents. More importantly, a session object refers to a part of the shared hypermedia workspace on which the session currently focuses; it keeps a list of actors working on the part for supporting group awareness; and it defines how the participants are working together (i.e., defining the degree of coupling between the browsers displaying the shared hypermedia space in the current session). Together, these two layers enable the provision of cooperative hypermedia. In our approach to coordination, we add a third layer on top that provides process support services (i.e. in the form of a cooperative activity space called process space).

### Activity Space

The foundation of this work is a hypermedia meta-model for representing task-specific hypermedia spaces, which we call activity spaces [24]. The basic elements of the meta-model are *links*, *nodes*, node content *pages*, and other *media objects*. The meta-model describes an activity space from three semantics: structural, relational, and computational. Structural semantics describe the graph type of a hypermedia structure. Along this dimension, links are classified into two categories: organizational and referential. Organizational links are constrained to form an acyclic graph, while referential ones have no such constraints. Relational semantics describe the relationship between typed hypermedia objects. Along this dimension, links and nodes are further classified into many application-specific semantic types, and the allowable relationships between them are specified. Computational semantics define task-specific actions or operations upon hypermedia objects.

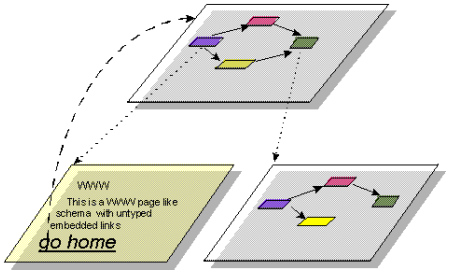


**Figure 1: Logical Structure of Activity Space**

The logical structure of an activity space is illustrated by Figure 1. Large squares are node content pages (composites), and small squares within pages are node representations, which in this model are anchors of embedded links that point to the content pages of the nodes. The straight arrow between node representations are independent links. The straight dotted arrow lines are embedded organizational links, and curved arrow lines are embedded referential links. Also, other multimedia objects can be included on a page. The shade under a page indicates that there is a page schema for each page. The structure of a composite is determined by its root node page schema, because the structures of its components are recursively defined along embedded organizational links. For details on activity spaces see also [24].

### Process Space

In this section we introduce the components for modeling processes. Abstractly speaking, both the logical structure of a workflow process and a hypermedia network can be seen as a directed graph. This makes the correspondence between many workflow concepts and hypermedia concepts straight forward:

- a graphical process definition language corresponds to the semantic net based hypertext "language",
- tasks (activities) correspond to nodes,
- control and data flow connectors between tasks correspond to links,
- processes (process-subprocess or task-subtask nesting structure) correspond to composites,
- process navigation corresponds to hypertext navigation,
- task and process definition corresponds to the definition of hypermedia schema and template, and
- process execution and process animation correspond to guided tours through a layered hypermedia graph consisting of composites.

The key difference between process structures and hypermedia structures is that process structures have their own computational semantics, such as task state transitions and flow dependencies, while hypermedia structures usually have no such semantics.

Our approach to process modeling is to integrate task and process computational semantics into the cooperative hypermedia activity space. As both the structural and relational semantics are directly applicable to process structures, what is needed is to incorporate task related semantics, such as state, time, and state transition semantics into hypertext nodes, and to incorporate control flow and data flow semantics into hypertext links. Thus, a process is represented as a set of task nodes connected by process links. They form a hypermedia composite consisting of potentially nested task nodes and process links among task nodes. An activity space for process definition and execution is then called a process space.

More specifically, in addition to the components identified in the integrated model and its activity space, the basic elements in a process space are described in the following:

- *Task nodes* describe task-specific information structures or logical steps within a process. Each task node has an associated page as its content which may also serve as a *container* of the task's input and output data,
- *Process links* represent precedence relationship between task nodes. They often associated with a data flow from the source task node to the target task node,
- *Transition conditions* are logical expressions associated with process links. They decide whether a process link will be selected or not, and
- *Pre- and post-conditions* are logical expressions which may be evaluated to see whether a task can be started or completed.

Ordinary hypermedia links, nodes, pages, and media objects are *application data*, which are application specific and usually not managed by a workflow system. However, in this model they are represented in a unified hypertext model, and can therefore be managed by a single system. The distinction of task nodes and process links from ordinary hypermedia nodes and links is reflected in the computational semantics attached to them. Application data can also be stored outside of the hypermedia system and accessed through *workflow relevant data* kept in task node attributes. The *workflow control data* – the data that relate to state, time, and the above mentioned *conditions* are represented as attributes of *task nodes* and *process links*. *Actors* are identified by an attribute of a task node.

In one incarnation of the process meta-model which is used as a default in our system, five task node types are identified:

- *simple task node* which represents an atomic task,
- *process task node* which models a subprocess,
- *automated task node* which represents a task to be performed by a computer program, and
- *iteration task node* which represents an iteration task or process.

As each task node has its own content page where application data can be stored and accessed, the process links serve as both control flow connectors and data flow connectors. For a simple task node, an automated task node, or an iteration simple task node, the node serves as its own input and output container. For a process task node or an iteration process task node, the input container is the start node of the process contained in the process task node and the output container is the end node of the process contained in the process task node. The AND-, OR-joints and the AND-, OR-splits of a process can be represented with a combination of the pre- and post-condition of task nodes and the transition conditions of process links. For instance, an AND-joint can be represented by a pre-condition which requires all transition conditions of the incoming process links to be true. The acyclic-constraint on process links and the organizational structure of nested task nodes does not exclude the possibility to model loops in a process. They can be represented with iteration task nodes.

Based on their associated dependency types or data flow patterns, process links can be classified into five types:

- *precede process link* which is a pure control flow connector, there is no data flow associated to it,

- *share process link* which specifies that the output container of its source task node shares the same content (page) with the input container of its target task node,
- *copy process link* which specifies that the input container of its target task node will contain a content copy from the output container of its source task node,
- *transfer process link* which specifies that the content of the output container of its source task node will be converted into the input container of its target task node according to predefined mapping rules, and
- *integrate process link* which specifies that the content of the output container of its source task node will be merged into the input container of its target task node.

The process computational semantics, such as those reflected in the above task node types and process link types, are assigned to typed hypermedia nodes and links at the process definition phase. They become part of the properties of the typed nodes and links to be readily used for creating process space instances.

#### Organization Manual and Agents

In this section we introduce the organizational components of the coordination model. *People, roles, agents*, and *organizational units* (projects and their teams) are represented in the basic CSCW support layer. Specific session types and session browsers are provided for them. The user-role relationship can be defined in a role definition browser. The organization structure and staff relationship (role-role association) can be represented with hypermedia structure and created in predefined activity spaces. The permission-role association can be defined on a per object or per object type basis. Agents are represented as deamon sessions running in the background. Agents can be managed with an agent browser, which can run agents, stop running agents, and modify or add new communication protocols (methods) for existing agents. One example of such agents is the mediator agent. The mediator agent registers all automatic tasks and checks periodically whether to trigger their state transition. Mapping the elements of the coordination model on cooperative hypermedia this way yields a shared environment for cooperative process modeling and execution. In the next two sections we discuss how groups may use such a cooperative hypermedia systems for cooperative process definition and execution.

#### Process Definition

A common collaborative authoring metaphor can be used for process definition. Users create process structure in the same way as ordinary hypermedia structures within a hypermedia authoring environment.

In CHIPS, a process definition corresponds to a process space schema, which can be created by sketching a process pattern (a template) as a hyperdocument. Such a schema contains the allowed components and the allowed relationships between these components in a process (or a document). Examples of such components are types of process links or task nodes as well as types of sub-processes or other hypermedia objects without workflow computational semantics. The component attributes, such as 'access rights' and 'actor', whose values would have global effect on all its instances, can be assigned at this stage.

After a process space schema (or template) is defined, a new process instance can simply be created by duplicating the template. This will lead to a completely initialized hypertext process structure that the users can then adapt to their needs and execute. However, it is also possible to create an instance directly from the schema. This will lead to an empty hypermedia workspace to start with. Users can then create their process structure and content on the fly. In any case, the constraints defined by the schema will be enforced. Thus, only valid process structures can be created. Users can also define a new schema from scratch or based on an existing schema, template or process instance. This is possible because every process instance can serve as a template as well as an instance to be executed.

#### Process Execution

The process execution mechanism can also be implemented in a hypermedia environment. Similar to InConcert [13], a process in our framework is a data object (a task node) not a program or script. The information on process state and conditions is distributed over task node instances, and state transitions are triggered and forwarded step by step along process links. This object-oriented feature allows a process instance to be modified by users (subject to their access rights) during the course of execution. This facilitates exception handling or flexible adaptation to new situations. Thus, both process definition and process execution be can done within one activity space.

To enact a process, first a process instance has to be created and attribute values that are local to this process instance, such as process start time adn task duration, have to be assigned. Then, the process can be started by an actor. When a process is started, the start task node is enabled and if it is an automated task node, its state will change to 'active' and the specified program is called. Otherwise, its human actors will be notified to activate the task node. A human actor can activate a task by opening the task node and setting its state to 'active', and perform the task in the activity space browser. Other actors of the task can enter the activity space at any time in its 'active' duration. When the task is finished, the actor(s) can change the state to 'finished', which in turn triggers the evaluation of its post-condition. If the post-condition is met, all its out-

going process links and their transition conditions are evaluated. Otherwise, the task has to be rescheduled for execution. In this case, if a deadline is reached, a message will be sent to notify the actors of the task and the mediator of the process. A process link is selected, if its transition condition evaluates to be true. The destination tasks of the selected process links are then evaluated. If their pre-conditions are met, notifications are sent to their actors. A process is considered terminated if all its end tasks are 'completed'.

## FLEXIBLE COORDINATION WITH CHIPS

In order to test our approach we built the CHIPS (Cooperative Hypermedia Integrated with Process Support) prototype. CHIPS provides different cooperative hypermedia session browsers to cooperatively define schemata for tasks and process structures and to execute processes. A short description of three of the browsers is given in the following. They are extensions of the browsers described in detail in [24].

A cooperative hypermedia *schema editor* allows users to define task or process schemata by crafting an example structure. After the example is created, users can create process (or document) instances by duplicating the example using a copy-as-template function. Another cooperative tool is the so-called *flexible editor* which switches off constraint checking and allows unconstrained manipulation of the hypermedia structure. This is an important feature since it facilitates the definition of emergent structure. For example, a process structure can be created gradually with continuous modifications, until a desired emergent process pattern appears. Then this process pattern can be turned into a process definition that is used to create new instances or to replace an existing process definition that is identified to be inadequate.

Finally, there is an *activity space browser* that is used to instantiate, modify, and enact processes. With it, users can navigate through the hypermedia network and activate tasks ready for enactment. Once a task node is enabled, the corresponding input objects are provided by the system and users can modify the content cooperatively. In addition, it is possible for users to switch back into process definition mode if they feel the necessity. They can modify the process structure, provided they have the necessary access permissions.

### Support for Informal Coordination

For direct communication, audio/video connections between participating workstations or sites can be set up to work with the CHIPS system. They provide people with an informal communication channel for social awareness and for applying social protocols to coordinate their work. Email connections can also be used for notifications and for transfer of external documents between actors. For indirect communication, shared hypermedia documents can be used as bulletin boards or as shared planning documents for team members. Annotations can be added in documents to express goals, make comments or explanations. In tightly-coupled mode, shared hypermedia can be used as a synchronous communication medium in distributed or co-located situations.

Activity space browsers also provide people with various workspace awareness: actors can always see (from the user interface) who is working on the same part of a document, and they can also activate a menu upon any visible objects to get more detailed information (attribute values of the objects), such as who is the creator and what is the site address of the creator. Actors can also use multiple tele-pointers to let their collaborators pinpoint the parts they are addressing. For higher-level overview information, actors can activate a menu upon the mediator image in the user interface to select a specific request, such as who is using the system and what part of the shared hypermedia space they are working on. They can also open the overview browser which displays an indented list view showing the process structure with indications of current users' positions and the current state of tasks. In addition to email, notification windows can also automatically open on receivers' desktops if they are on-line.

### Support for Variable Coordination

The CHIPS system can provide an end-user-controlled amount of automatic coordination for a wide range of tasks, from unstructured to strictly structured ones. This is enabled by our process space structure that consists of nested nodes with various page types (schemata of sub activity spaces). The system allows manual tasks to include subtasks that are automatic tasks or processes, and allows tasks whose subtasks have no flow dependencies. This is the kind of combination not available in current workflow systems. Another distinct flexibility of the system is that ordinary hypertext structure can be turned into process structure, and process structure can be changed back into ordinary hypertext structure. This flexibility facilitates the development of emergent process patterns.

The dual interpretation of hypermedia structure as process structure and as the information that people manipulate when cooperatively executing the process makes it easier to switch between process definition and process execution. As a consequence, exceptions and adaptation to changing processes become easier to handle.

The process space schema and its example-based definition method offer another distinctive feature of the CHIPS system. These schemata can be incrementally changed to meet the changing needs of their process space instances. The example-based definition method

allows ordinary users to define new process patterns. Moreover the system can make use of schemata to maintain the structural and relational consistency of a process structure. It can provide intelligent aid in creating task and process structures by providing allowable choices to prevent violations, and issuing warnings (and aborting the violating operation) for violations that can not be foreseen before an action is taken.

### An Application Example

This example shows how a decision making task is performed in a *cooperative process*. The sample task is to decide whether to use Java or Smalltalk for the new projects of a research group. An issue-based decision making scheme is used in a *multi-phase process*. In this section, first the creation of the process schema and role-based organizational context is briefly described. Then the creation, execution, and modification of the process instance is presented.

To initiate a process definition, first, roles taking part in the process are defined with a *user-role assignment* editor (if existing roles are not applicable to this process). In this case, the roles are 'software issue mediator' and 'software issue advisor'. The first role mediates the process and is taken by the head of a research group, Haake. The second role is taken by all the decision-making participants in the research group. Then, a decision making *process schema* (named Co-decision) is created, which includes two subtask schemas for the two phases of the process. The first phase is a planning phase where Issues and Positions are raised. The second phase is a discussion phase where advisors would debate by enumerating pros and cons to the positions generated in the first phase. Then, in a *permission-role assignment* dialog box, access permissions are assigned to roles on a per object type basis. In this case, the authorized roles are 'software issue mediator', 'software issue advisor', and 'any user'. More specifically, Issue type nodes can be updated (created/deleted) by a mediator. Position, Pro and Con type nodes can be updated by all advisors. Only the mediator can authorize permissions and execute the process (i.e., trigger state transitions. If the phases are defined as automatic tasks, then the transition is triggered automatically). 'Any user' can query (navigate) the process, but not change it. The first two roles are then assigned as actors of the process (and its phases) in a *role-actor assignment* dialog.

Then, a *process instance* (see Figure 2) of the schema is created using a copy-as-template operation. Attribute values, such as process 'start time' and task 'duration', whose scope is local to this process instance, are assigned. Next, Haake started the process by triggering the play button on the user interface of the process space. At this point, Haake realized that there


Figure 2: A process space for co-decision making

was something missing in the process definition: a ballot should be performed after the discussion phase. By consulting the *CHIPS system mediator*, he noticed that user Wang was on-line. Thus, he invited Wang to discuss the modification of the process by adding Wang to the user list of his current editing session. Working together they *modified the process structure (the schema and the process instance) on-the-fly*. They added a Ballot phase, which was defined as a shared whiteboard for carrying out the task with *informal coordination*. Then, they added task computational semantics to the Ballot node type and *turned the regular hypertext node into a task node*, so that task related attributes (such as 'duration' and 'actor') could be assigned. They also added an annotation on the root page of the process instance *as an indirect communication mechanism* to *make the goal of the process clear*. Also, an Exploration node and an explore link were added as *associated references* to background information of the primary co-decision task.


**Figure 3: Structure created at the first step**

Figure 2 also explains the user interface layout of the activity space browser. Each instance of a node is represented as a box, which carries its name and its type as a little label. Links are represented as arrows carrying

their type name as a label, too. Color coding is used to signal the state of a task node, e.g. green task nodes are active. The node type label is also used to display planned task duration and end date as well as the logic of pre-conditions for task nodes. Here,'&' means AND-joint. Similarly, the letter 't' (for true) at the end of link labels denotes the value of the transition condition of the link. Figure 3 shows Haake working on the first step of the process. After an Issue node and two nodes for two possible positions were created, he pressed the Stop button to end the task. After the task was completed, its *data were passed along 'integrate' link* into its following task node, and all discussion advisors were notified by an *email notification*.

Figure 4 shows the extended content of the following task node. The three nodes at the top were passed from the previous phase as references (to the same pages). If a 'copy' link rather then the 'integrate' link is used, then the passed nodes (and their pages) would have been copied. The bottom two nodes are extended in this phase. The user list shows that Wang, who opened this node, is now working alone on this page. Since the session he works in is in *loosely-coupled cooperation mode*, his navigation action would not affect the browsers of cooperating users. However, if users decide to work in a *tightly-coupled cooperation mode*, all users would follow the navigational actions of other users in the same session. Similarly, after the discussion task, all the actors of the final phase are notified to cast their ballot. For process consistency and *record-keeping* purposes, when a task is completed, the task node is frozen to prevent further changes. Because all the information on the decision making process is kept in the shared hypermedia space, new group members or other people can navigate the space to see how this decision was made and what the final decision is. Also, the process template can be reused for other similar decision making tasks.

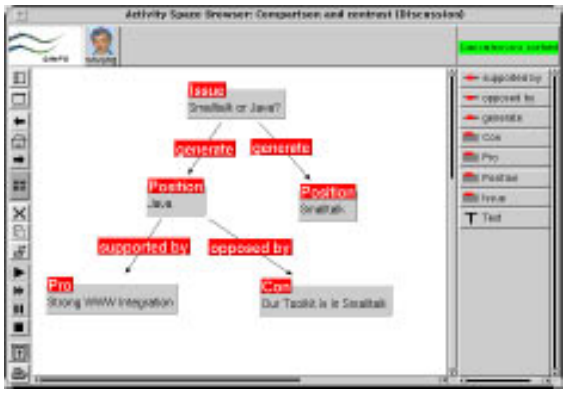


**Figure 4: Structure extended at the second step**

This example shows that CHIPS can not only support team members to define a structure, but can also support them to create such a structure step by step cooperatively. The 'flow' and 'fit' dependencies are primarily handled with a workflow approach and the 'sharing' dependencies are primarily handled with a groupware approach, i.e., using concurrency control and role-based access control mechanisms. The combination of the two, together with the flexible hypertext information structure, makes it possible to support flexible coordination. To see the time-based nature of the example better, a series of pictures is given at our web site (http://www.darmstadt.gmd.de/concert under activities/internal projects/CHIPS/example). Readers with a PC can access a ScreenCam movie at the CHIPS web page to see CHIPS at work.

## CONCLUSION

Flexible coordination is needed for different work situations and work dynamics. Using cooperative hypermedia to uniformly model all objects that represent coordination mediums and shared artifacts makes it easier to switch between formal and informal coordination mechanisms. The process space representation, especially the composite structure of nested task nodes with different page types, supports variable degrees of automated coordination for tasks that are composed of parts (subtasks) requiring different coordination mechanisms. The possibility to transform between hypermedia objects with process computational semantics and hypermedia objects without process computational semantic facilitates the development of emergent process structures and the adaptation of existing process structures. These unique features make the cooperative hypermedia based approach to flexible coordination a very promising one to bridge the gap between informal and formal coordination mechanisms.

Compared to previous work, our approach to flexible coordination adds explicit coordination support (work process support) to cooperative hypermedia systems. Unlike many workflow systems, our approach facilitates (1) the integration of processes with different degrees of automated coordination support, and (2) the use of informal communication and cooperation functionality as provided by the cooperative hypermedia paradigm.

The possibility of incorporating process computational semantics into hypermedia structure opens many new application areas for cooperative hypermedia systems. We are currently exploring using such hypermedia systems to support new organizational forms that emphasize processes and team work, to support the coordination of off-line tasks of mobile workers, to support task-based version management [6], and to support hypermedia authoring for multimedia presentation. This kind of hypermedia system may also be used as a coordinating middleware to integrate legacy applications

into a well-coordinated cooperative process.

**ACKNOWLEDGEMENTS**

The authors especially thank Christian Schuckmann, Dan Russell, and the anonymous reviewers for their detailed suggestions. Many thanks to Ajit Bapat, Jennifer Beck-Wilson, Daniel Tietze and Martin Wessner for commenting on earlier drafts of this paper.

**REFERENCES**

1. Akscyn, R., McCracken, D., and Yoder, E. KMS: A distributed hypermedia system for managing knowledge in organizations. *Commu. of the ACM, 31, 7* (1988), 820–835.

2. Ellis, C., and Nutt, G. Modeling and enactment of workflow systems. *Advances in Petri Nets 93* (June 1993), 1–16.

3. Ellis, C., and Nutt, G. Multi-dimensional workflow. In *Proc. of the IDPT'96* (1996).

4. Furuta, R., and Stotts, D. Interpreted collaboration protocols and their use in groupware prototyping. In *Proc. of ACM CSCW'94* (Oct., 1994), pp. 121–313.

5. Gronbaek, K., Hem, J. A., Madsen, O. L., and Sloth, L. Designing Dexter-based cooperative hypermedia systems. In *Proc. of ACM Hypertext'93* (1993), Papers, pp. 25–38.

6. Haake, A., and Hicks, D. Verse: Towards hypertext versioning styles. In *Proc. of ACM Hypertext'96* (1996), pp. 224–234.

7. Halasz, F., and Schwartz, M. The dexter hypertext reference model. *Commu. of the ACM, 37, 2* (Feb., 1994), 30–39.

8. Harada, K., Tanaka, E., Ogawa, R., and Hara, Y. Anecdote: A multimedia storyboarding system with seamless authoring support. In *Proc. of ACM Multimedia'96* (Nov., 1996), pp. 341–351.

9. Hollingsworth, D. *The Workflow Reference Model.* Workflow Management Coalition, TC00-1003, Dec. 1994.

10. Malone, T., and Crowston, K. What is coordination theory and how can it help design cooperative work systems? In *Proc. of CSCW'90* (1990), pp. 357–370.

11. Malone, T. W., and Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys.* (1994).

12. Marshall, C., and Rogers, R. Two years before the mist: Experiences with Aquanet. In *Proc. of ACM Hypertext'92* (1992), pp. 53–62.

13. McCarthy, D., and Sarin, S. Workflow and transactions in InConcert. *IEEE Data Engineering, 16,2* (June 1993), 53–56.

14. Nutt, G. The evolution towards flexible workflow systems. *Distributed Systems Engineering Journal, Special Issue On Workflow Systems, 3, 4* (Dec., 1996), 276–294.

15. Scheer, A. W. *Business Process Reengineering: Reference Models for Industrial Enterprises (2nd ed).* Springer-Verlag, New York, 1994.

16. Schlichter, J., Koch, M., and Buerger, M. Workspace awareness for distributed teams. *Lecture Notes on Computer Science* (1997), Springer.

17. Schuckmann, C., Schümmer, J., Kirchner, L., and Haake, J. Designing object-oriented synchronours groupware with COAST. In *Proc. of ACM CSCW'96* (Nov., 1996), pp. 30–38.

18. Sheth, A. State-of-the-art and future directions. In *Proc. of NSF Workshop on Workflow and Process Automation in Information Systems* (May 1996).

19. Shipman, III, F. M., Chaney, R. J., and Gorry, G. A. Distributed hypertext for collaborative research: The virtual notebook system. In *Proc. of ACM Hypertext'89* (1989), pp. 129–135.

20. Smith, H. T., Hennessy, P. A., and Lunt, G. A. The activity model environment: an object-oriented framework for describing organizational communication. In *Proc. of first ECSCW* (Sept., 1989), pp. 160–172.

21. Smith, J. B., and Smith, F. D. ABC: A hypertext system for artifact-based collaboration. In *Proc. of Hypertext'91* (Dec., 1991), pp. 179–192.

22. Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Schutt, H., and Thüring, M. SEPIA: A cooperative hypermedia authoring environment. In *Proc. of ACM Hypertext'92* (1992), pp. 11–22.

23. Trigg, R., Suchman, L., and Halasz, F. Supporting collaboration in Notecards. In *Proc. of CSCW'86* (Dec., 1986), pp. 1–10.

24. Wang, W., and Haake, J. Supporting user-defined activity spaces. In *Proc. of ACM Hypertext'97* (Apr., 1997), pp. 112–123.

25. Wiil, U., and Leggett, J. Workspaces: The hyperdisco approach to internet distribution. In *Proc. of ACM Hypertext 97* (Apr., 1997), pp. 13–23.