Proceedings of the Third European Conference on Computer-Supported Cooperative Work
13-17 September, 1993, Milan, Italy
G. De Michelis, C. Simone and K. Schmidt (Editors)

Sharing To-Do Lists with a Distributed Task Manager

Thomas Kreifelts, Elke Hinrichs, Gerd Woetzel
Gesellschaft für Mathematik und Datenverarbeitung, Germany

Abstract: We describe a simple and powerful tool for the management of distributed work: the Task Manager. Common tasks may be shared and manipulated independently by a number of people. They are represented as shared to-do lists at the user interface. With the help of the tool, users may organize cooperative tasks, monitor their progress, share documents and services, and exchange messages during task performance. The paper gives the motivation for the development of the Task Manager, implementation details, and a first assessment of its usefulness.

1 Introduction, or: Why do we need tools for the management of distributed work

Working in large and geographically distributed business organizations and government agencies requires that individuals and groups at different sites engage in intensive cooperative activity. Business teams are formed from different parts of an organization, and agencies in non-centralized government settings have to cooperate over large distances. There is a trend towards formation of joint ventures and consortia to carry out large projects across organizational and national boundaries. All this leads to an increasing need for computer tools to support distributed task management in order to provide better overview and to avoid expensive inefficiency, errors, and delay.

Another incentive for the development of support tools for distributed task management is the emergence of alternative forms of the organization of labour: on the one hand we have an increasing number of people working "on the move", e.g. at a customer or on a construction site. On the other hand we observe a tendency to-

wards people working at home in order to avoid commuter traffic and enable more effective part-time working.

The EuroCoOp project set out to provide the required support by developing facilities for distributed work management, including tools for the scheduling and coordination of cooperative activities, progress monitoring, formal reporting, and joint authoring of documents. Within this project, over the period of the past two years, we have developed the Task Manager as a tool to share and manage common tasks in a community of distributed users[1].

The Task Manager is a prototype software system for specifying and managing cooperative activity. With its help, users may organize (create, refine, and modify) cooperative tasks, monitor their progress, share documents and services, and exchange messages during task performance. The Task Manager distributes task specifications, attached documents, and messages to the involved users in a consistent way. It is meant to support the management of work distributed in time and/or space by providing

- support of organization and planning of collaborative work (who does what, with whom, until when, using what?),
- up-to-date overview of collaborative activity and work progress,
- dynamic modification of work plans during performance,
- availability and exchange of documents and messages within groups of people involved in task performance.

The Task Manager organizes distributed work in tasks which have a person responsible, a deadline, other participants, the material necessary for task performance, and possibly subtasks. The primary user interface of the Task Manager is a hierarchically structured to-do list which displays all tasks a user is involved in and which allows direct access to the relevant information attached to a task.

2 Motivation, or: Why develop coordination tools

There has been considerable research on coordination tools; recently also a number of commercial products has appeared claiming to support coordination.

Commercially available products come in different flavours: some cover certain aspects of the problem like personal productivity tools, group calendars for local networks, project management systems (which are single user applications for project managers). Others represent closed "groupware" solutions with no clear concept of, and not particularly tuned to, coordination. The integration of, or interoperability with, existing computer support (like word processors, data base sys-

[1] Research reported in this paper has partly been funded by the Commission of the European Communities within the project no. 5303 "EuroCoOp - IT Support for Distributed Cooperative Work" of the ESPRIT Programme in 1991 and 1992. Project partners were TA Triumph-Adler, Århus Universitet, BNR Europe, empirica, GMD, Jydsk Telefon, NeXor, and Storebæltsforbindelsen.

tems, spreadsheets etc.) usually presents a serious problem. Consequently, there is not much support available on the market that provides the comprehensive coordination functionality and the necessary amount of "openness" to third-party software that we consider essential.

Within academic research, quite a number of models, prototypes, and systems have been developed with the explicit goal of coordination. Not many of these approaches have ever been implemented or even put to use; so there is not much experience with coordination tools in spite of the obvious need for such tools. The few experiences reported on computer support for coordination exhibited a number of difficulties with those systems, particularly the lack of flexibility and interoperability. This has also turned out to be true for the office procedure system DOMINO (Kreifelts et al. 1991) that we had implemented and assessed. Consequently, we focused on two problems:

- the *rigidity* of pre-defined procedures and imposed structures which lead to a limited application domain and non-adequate exception handling in a number of situations
- the *isolation* from informal communication, information sharing, and other forms of computer support.

To avoid the implications of rigidity, it has been argued in (Hennessy, Kreifelts and Ehrlich 1992) that future coordination support systems should focus on Schmidt's proposal (1991) of treating models of cooperative work as *resources* to be defined, modified, and referred to for information purposes instead of as *prescriptions* to be adhered to. To overcome the relative isolation future coordination support systems have to be able to interface to existing computer systems that support the actual work — coordination is never an end in itself.

Another aspect that so far seems to have been largely neglected is the effort needed to make use of coordination systems: most systems require *pre-organization* of the cooperative work by some sort of systems administrator before a system may be put to use by an ordinary user. Instead, one would like to have coordination systems that encourage *self-organization* of cooperative work by the end-users themselves. In order to overcome this initial barrier of using coordination systems, the genericity and simplicity of the underlying coordination model are of primary concern.

3 Description, or: What are the basic concepts of the Task Manager

We now describe the framework on which the Task Manager is based: its components and their attributes, and the operations to create and modify task structures. We then give a picture of how all of this is presented at the user interface.

3.1 Components

The central notion of the Task Manager is that of a *task*. In order to perform a task, *people* use shared *documents* and/or *services* and communicate by sending *messages* to each other.

- Tasks
  The Task Manager's notion of a task has various aspects: one could think of a task as of a project, i.e. a common goal of a set of people (*result-oriented*). A task may be broken up into several sub-tasks and dependencies may be defined between them and their documents. The more detailed specifications are given, the more a task resembles an office procedure with causal dependencies between subtasks and documents of a task (*procedure-oriented*). A task may also be used as a simple folder with little or no structure defined: in that case a task is simply a shared container of subtasks, documents and/or services, and messages that people exchange about in a common task (*information-sharing-oriented*).
- Documents/services
  In everyday office life, there are generally resources of some sort needed in order to achieve the goal of a task; therefore, each task may have resources attached to it. Resources are "pointers" to various kinds of computerized objects: the first kind of resource one could think of would of course be documents that are shared between the participants of a task. But there are also other resources like rooms, budgets, machinery, etc. that may be crucial to the performance of a common task. In our system those resources are handled by services that are implemented outside of the actual Task Manager, but may be referred to and shared by the participants of a task.
- People/users
  The most important "resource" are the people involved in a common task. We distinguish between various levels of participation and of competence. There is a set of people involved in a task, the *participants*, that all have equal access rights to the attributes of a task, its documents and services and its messages. Participants may invite other people to take part in the task, i.e. to become new participants or observers. *Observers* are people interested in the completion of the task with read access only to any information and the right to participate in the informal message exchange associated with the task.
  One of the persons involved in a task is "more equal" than the others: the *person responsible* for the performance and the outcome of the task. S/he has exclusive write access to some of the tasks attributes, e.g. state, start date and deadline, and only s/he may reassign the responsibility of the task to another person.
- Messaging
  Participants and observers may freely exchange informal electronic mail messages within the context of a task. By integrating and facilitating extensive mes-

sage exchange we give more room to flexible social protocols in contrast to regulations dictated by the system.

3.2 Attributes

Tasks, participants, documents, and services are specified in more detail by a set of attributes each. The *title* of a task both identifies a task to its participants and gives a short and concise description of its goal. The title of a task is the only mandatory attribute — thus, the user is not forced to fill out long forms before s/he can actually start working on a task. All other attributes are either optional or set to a default value by the system. E.g., when a user initially creates a task s/he automatically is the *person responsible* for it.

As mentioned above, the person responsible for a task has special rights; in particular s/he may set the *state* of a task. We only distinguish between the completion state of *not finished* and *finished.* This state is set by the user. Apart from the completion state, a task can be *pending*, i.e. there is a causal dependence on another task not yet finished, or *not pending*. This state is set by the system automatically, and should help the user decide when best to start with the actual work on a task. As a third kind of state information, a task can be *acknowledged* by the responsible actor. This is to inform the co-workers of the responsible person's awareness and acceptance of the task s/he has been assigned to.

Another important attribute is the envisioned *deadline* of a task. The system reminds the user of approaching deadlines, but does not enforce any actions with respect to overdue tasks.

Besides those most important attributes there are a number of other attributes that allow the user to specify in more detail how a task should be performed: time-related data, such as *start date*, data that describes causal *dependencies* between tasks and between tasks and documents, and *personal data* attached to a task, such as notes etc. The latter attributes are purely local and are not distributed to and shared by the other participants.

Documents and/or services may be attached to any task in which the user participates at any time. They consist of a name, a history of who did what and when, the owner of the document and other data. After discussions with prospective users we added the *abstract* attribute of a document: it contains an informal text description of the document and it frees the user of having to transfer, open and read the entire document when s/he is only interested in a resume. For reasons of simplicity, we decided to implement a semi-transparent file transfer service (cf. section 5). Other resources as mentioned above are handled by separately implemented services; the system keeps an account of so-called service requests, but leaves other details to the respective service.

Participants are people that work together on a common task. They are worldwide and uniquely determined by a pair of ids or by an X.500 distinguished name.

We also support more user friendly names, individual address books, and access to the X.500 directory service.

3.3 Operations

Users can create tasks and subtasks, dependencies between tasks and between tasks and documents, they may set and modify attributes, add, modify, and remove documents and service requests. Persons responsible for a task may refuse responsibility and they may reassign it to another user. Any participant can introduce new participants or observers to a task. Tasks may be copied and pasted or moved around freely. All of this may be done at any time, thus allowing dynamic modification of the work situation at run-time.

Basically, the system distributes information on tasks, makes available resources across the (world-wide) network, keeps the data up-to-date, and resolves conflicts of synchronization. Each user has instant access to the shared tasks s/he is involved in. The system guarantees a consistent view on tasks for each participant. Additionally, the system keeps track of the actions the users take. Thereby monitoring and task tracking at execution time as well as report generation after completion of a task is rendered possible.

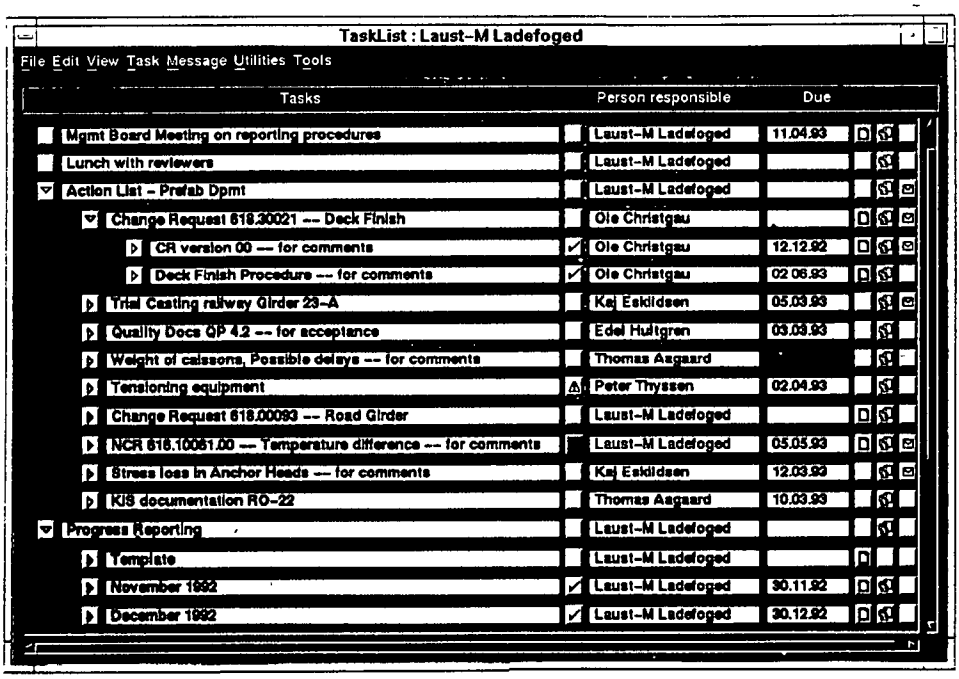

Figure 1. Task List user interface

3.4 User interface

The set of tasks a user participates in is presented at user interface level as a *Task List*, very similar to conventional outliner programs (Figure 1). The Task List gives an overview of the hierarchically ordered tasks along with a condensed view of the most important attributes: *title*, *person responsible*, *deadline*, *documents/services*, *participants*, *state*, and a list of *messages* that have been exchanged within that task. Operations are invoked by selecting menu commands and/or by directly typing in the attribute fields. In most cases, the Task List will suffice to display the information and to perform the necessary operations.

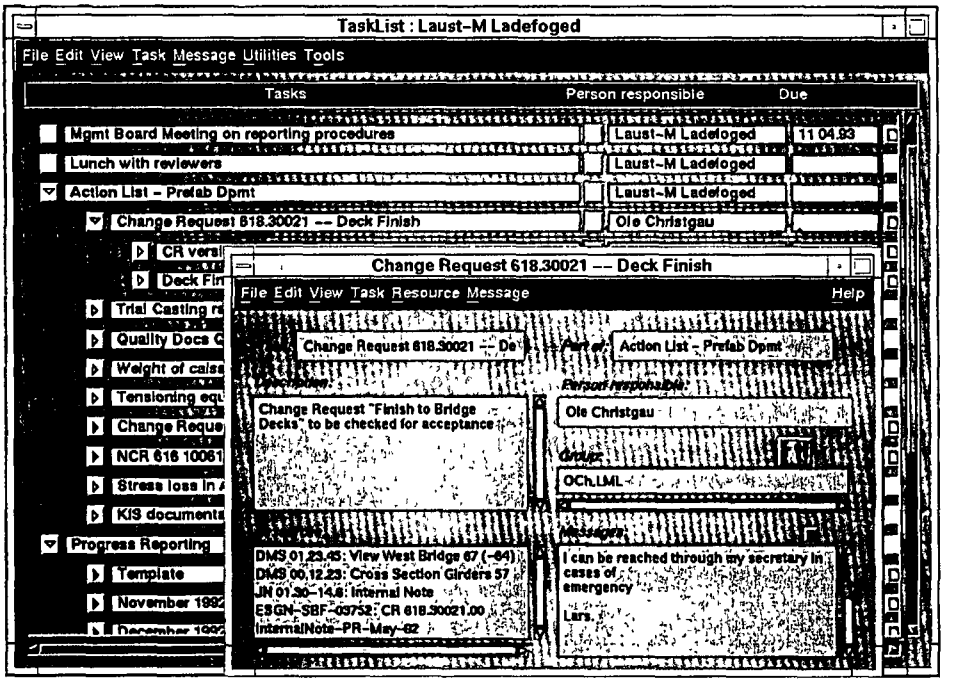

Figure 2. Task Editor user interface

For more detailed information, we provide a form-oriented *Task Editor*; it may be used to view and edit in detail all of the attributes (Figure 2). The Task Editor also provides access to a local address book as well as to external addressing information such as the X.500 directory.

The *Dependency Editor* represents groups of sub-tasks and documents in a net-like way; it mainly serves the purpose of graphically displaying and editing dependencies between tasks and resources. The dependency structures are similar to those used in workflow systems, but are interpreted differently: instead of driving a workflow with a strict sequencing of steps, dependencies in the Task Manager represent recommendations which may be changed or overridden by the users at any time.

The *Logging Tool* gives a complete history of events with respect to all or to selected tasks, and lets the user acknowledge new events (Figure 3). Thereby, s/he may find out what's new since "last Tuesday".

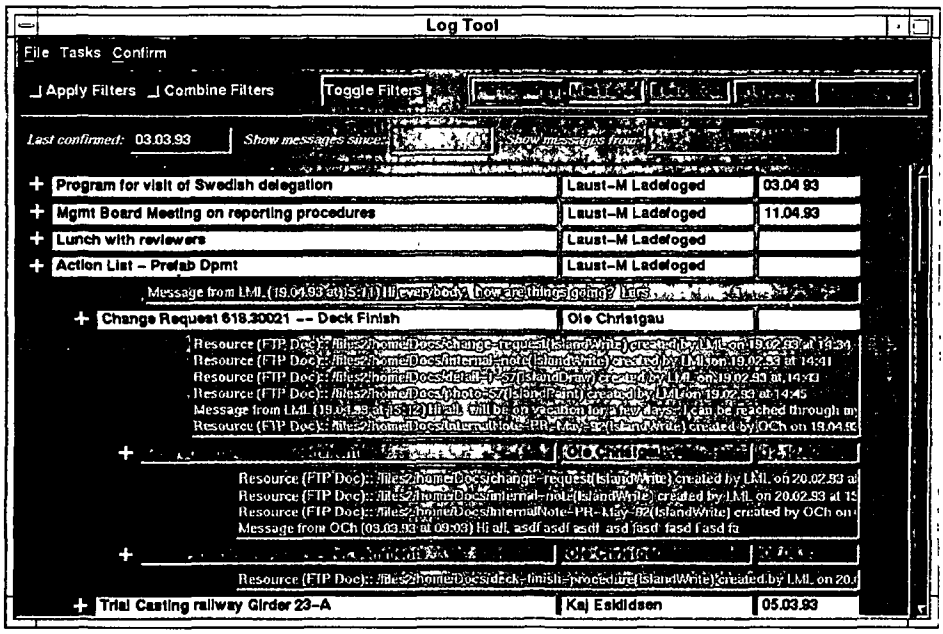

Figure 3. Logging Tool user interface

Other views are possible and desirable, but are not yet implemented: filters on the Task List on the basis of selection criteria, sorting according to other criteria, Gantt or milestone charts for a group of sub-tasks in order to facilitate time management, etc.

4 Usage, or: What can you do with the Task Manager

In the following, we describe in more detail how we envisage the usage of the Task Manager — this will also include other forms of system usage not generally associated with the term "task coordination".

4.1 Different Kinds of Usage

We have chosen some typical forms of usage which will demonstrate the versatility of the Task Manager tool; like with ordinary paper to-do lists, such different types of usage may be mixed with one another.

- Personal To-Do List
  This is the simplest form of use: tasks in the Task Manager's task list are not shared with other users, but serve as reminders for actions to be taken; that may range from trivial tasks simply represented by their title to specifications of complex projects. Entries in the task list may also simply represent "folders" for a set of tasks.
- Brainstorming, Conferencing
  A task may be thought of as an environment for off-line brainstorming. Prospective contributors are added as participants to the task and offered a short description as to the purpose of this activity. They may add their ideas as text messages or documents of any kind. The person responsible acts as a moderator and may form subconferences if necessary. A passive observer status is possible.
- Meeting Preparation
  The preparation of a meeting can be made a task: the invitation and other important documents will be distributed to the participants as documents attached to such a "meeting task", the participants can give their feedback via the task conference, the organization of the meeting, the writing of the minutes or other follow-up activities can be made subtasks of the meeting task.
- Project Planning and Monitoring
  The support of this activity is a natural for the Task Manager: it supports a gradually and dynamically refinable and restructurable structure of tasks and subtasks, the responsibility may be assigned and reassigned, the completion state is reported by the responsible persons as tasks are carried out, and deadlines can be set and monitored.
- Project Execution
  Also the project work itself can be coordinated, because relevant material and applications may be attached to the respective tasks and subtasks of the project in order to be shared and worked upon by the project members. The somewhat primitive document access and version control mechanisms would have to be complemented for more sophisticated applications by special services.
- Repetitive Tasks
  The Task Manager supports the reuse of task specifications which may be first edited to fit the current situation and then "pasted" into the list of tasks. Templates for repetitive tasks may be stored privately or copied from organizational databases.
- Bulletin Board
  This again is a very "simple" use of the Task Manager: a task represents a topic described by the title, the bulletins are added as messages or documents to the task and are then available to all participants. Hierarchical structuring of topics is possible via creating subtasks. New participants may be introduced by existing ones. Participants no longer interested may leave the topic in question.

The above is not meant to prove that almost every kind of collaborative (or even non-collaborative) activity may be supported by the Task Manager; of course we are aware of collaborative tasks which would require more specialized support, like e.g., the joint authoring of documents. We want to show that the Task Manager can be used in various ways not just for the management of clearly defined tasks.

The various kinds of usage are not necessarily to be kept separate from one another but can dynamically develop from one kind to another. For instance, a task in the personal to-do list ("Should develop a productivity tool for our software production") can be turned into a brainstorming item by adding a few more participants. Documents attached to such a personal to-do list item, e.g. a draft proposal ("A Distributed Task Manager for Software Production") are distributed to the other participants and may now be commented on. Eventually, a project plan could develop from this activity; some persons involved in the planning activity drop out of the task, new participants are introduced which have special skills needed for the project. A senior manager is added as an observer of the top level task so as to be informed of the essentials of the project, and so on.

4.2 Synchronous and Colocated Usage

Within the well known two-by-two matrix of CSCW systems introduced by R. Johansen which distinguishes systems along the dimensions of temporal and spatial distribution, the Task Manager clearly falls into the asynchronous, geographically distributed category: the Task Manager does not require that its users sit in front of their workstations at the same time nor in the same room. On the other hand, it does not forbid this, i.e. the Task Manager may also be used in the colocated or synchronous case.

For instance, a user may realize that another user is actively manipulating tasks around the same time s/he does and could react by sending a message or even call the other participant and discuss rearranging or rescheduling a group of subtasks while both would look at the task list and do some editing. They could also sit in front of the same display and work together on some tasks; the results of such a session would be automatically distributed to each of them (and any other participants).

Another example is that of a project meeting where the task structure is discussed by the participants equipped with the Task Manager on mobile computers. So, while mainly meant for work situations distributed in time and space, the Task Manager may also be used synchronously or in colocated situations.

4.3 Scalability of Use

The Task Manager may not only be used by a limited number of users over a local network. Special attention has been paid during its development to its scalability with regard to the number of users, the number of tasks, and the dimensions of

geographical distribution. The Task Manager is not limited in this respect other than by the availability of computer storage and store-and-forward communication facilities. It may be used for the support of large user communities over large geographical distances without restrictions or deterioration of functionality.

5 Implementation, or: How does the Task Manager work

Starting with the conception of the coordination model, the Task Manager was developed over a period of two years, with a first prototype after about 15 months. The subsequent phase of evolutionary development included: stabilization of the prototype; evaluation in simulated work situations by potential users which brought valuable feedback to the developers; essential enhancements of functionality and user interface towards a workable system.

Having had an early working prototype paid in terms of quality of the present system. Now that the Task Manager is rather stable it is in use within the developers' group, among other things for project management and the development of the next version of the system itself. We plan to gradually extend this system usage to different types of settings, e.g. with the partners of our present European project.

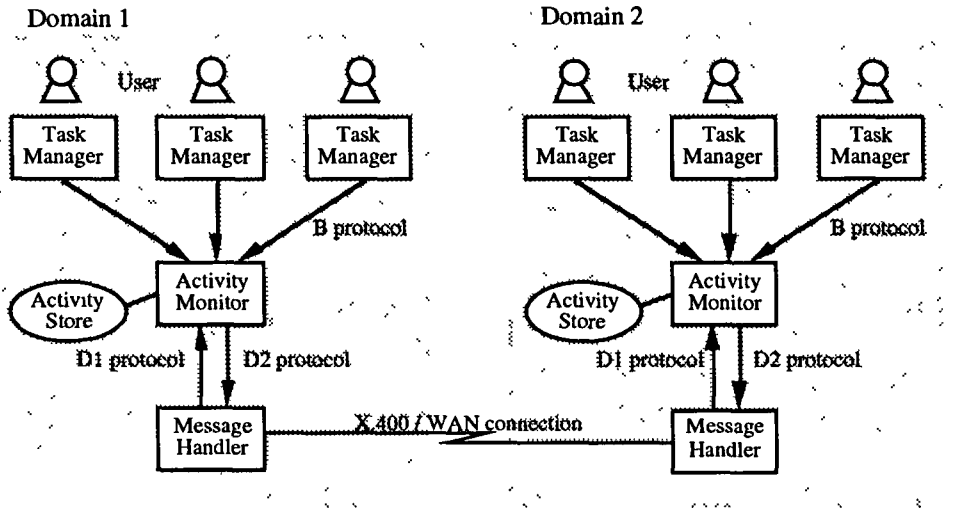

Figure 4. Software Architecture of the Task Manager

Figure 4 gives an outline of the software architecture. It shows two domains linked by standard X.400 store-and-forward message transfer — however, the number of domains is not restricted. The concept of domains reflects the different speed of

LAN or WAN communication. Within a domain, a client-server approach is used to update a shared task structure, while X.400 messages are used to distribute changes to other domains. Every user of the system is located in a specific domain and uses his/her own instance of the Task Manager. This tool manages user specific views of the user's task list. Using remote procedure calls, the Task Manager talks to the Activity Monitor to get up-to-date data and to request changes to the domain-wide shared Activity Store. In each domain, there is one instance of the Activity Monitor that serves multiple Task Manager instances.

When a user modifies a specific task or the task structure, the Task Manager sends an operation request to the Activity Monitor. The monitor executes the operation in the Activity Store and broadcasts the request to the remote domains involved via the Message Handler. When an operation request arrives from a remote domain, the Message Handler passes it over to the Activity Monitor which then updates the corresponding task objects in the Activity Store accordingly.

The task objects are actually replicated in different domains. This is needed because user operations on tasks should be immediately reflected in local changes. A user will not wait for operation requests or results transported over WAN connections, but requests a fast response — even if the response is preliminary in some cases as discussed below. Strictly speaking, tasks have to be replicated not only in those domains where participants of the task itself are located but also in domains where participants of supertasks can observe a task (a task can be subtask of more than one supertask). Therefore, the replication of task objects depends on task participants and the hierarchical task structure which both may change dynamically. Task replication is completely transparent at the user interface. The essential purpose of the Activity Monitors is exactly the distributed control of the replication and modification of task objects in spite of WAN connection breakdowns between Activity Monitors and lost or duplicated messages on WAN connections.

Obviously, a concurrency problem has to be solved for this architecture: operation requests arrive asynchronously at the Activity Monitors from local and remote domains, while all users in their different domains should eventually have a compatible view of the task structure (which needs not necessarily be the same view). As a solution, we have designed the D protocols between the Activity Monitor and the Message Handler in such a way that the causal order of operations on task objects is preserved[2]. As a main feature, our protocols enable the Activity Monitor to detect concurrent updates of task attributes, and to resolve conflicting assignments by assuming a linear order on those events.

Of course, all Activity Monitors must reach the same belief about the linear order eventually. The effect of this strategy for users is that sometimes local changes be-

[2] Our approach is similar to the *cbcast* method within process groups in the ISIS system (Birman, Schiper and Stephenson 1991). But in contrast to the ISIS system, we only need causal order to deal with changes of a tasks' participant set, such changes are not atomic (membership changes of process groups in ISIS are atomic).

come "overwritten" by changes due to remote participants. Because all changes are highlighted at the user interface, a user at least gets aware of what has happened.

The asynchronous distribution mechanism for tasks is also used for the distribution of messages, documents and services. Lists of messages, document references, and service access points are attached to tasks and technically managed as task attributes. In order to attach or to change a document, the document or new version is copied with a new filename to a host in the local domain, and the pair "host/filename" is distributed. When accessed remotely, the documents are ftp'ed (a cache can be used on the remote site to avoid multiple remote copies).

While the distribution of new versions of documents admits only an asynchronous, coarse-grain update of documents, the usage of services allow for distributed editing and synchronous communication. Task Manager and service use a simple protocol to agree upon what X server the service is to be used when it eventually contacts the user. All services integrated in our system must obey this start-up protocol.

The concept of documents and services renders the system "open-ended". We support documents of any kind as long as their associated tools, like word-processors or graphical editors, run on the underlying UNIX operating system.

The Activity Monitor is implemented as an ISO-ROS service (Rose, Onions and Robbins 1991) supporting two access protocols B and D_1. The Message Handler offers another ROS service: via the D_2 protocol the Activity Monitor propagates its data across domains. The protocols are defined in ASN.1 (Steedman 1990). All clients and services of the system are implemented in C++ using ISODE (Rose, Onions and Robbins 1991).

6 Evaluation, or: Does the Task Manager live up to the expectations

Here we report on first assessments of the usability and usefulness of the Task Manager. First, we take a look at our design goals.

Flexibility : we have already shown that the Task Manager is a versatile tool that may be useful in a variety of work situations. Tasks may be performed by just one user or shared by any number of users in the same way, a shared task may be structured to various degrees of refinement and causal dependency, and all attributes, including the participating users, may be changed dynamically.

The only restriction to flexibility perceived so far is the rather egalitarian model of task sharing: all participants share the same view. Extensions to the tool might be needed in order to adapt to rather different organizational cultures than we originally had in mind.

Interoperability: the concept of attaching documents and services to tasks opens up various ways in which existing tools or applications to be created may interoperate with the Task Manager. First, arbitrary documents attached to tasks

may be opened and worked on without leaving the task management context. Secondly, services offer specific cooperation support not covered by the Task Manager within the task management context.

Self-Organizability: users of the Task Manager may organize their collaboration as they see fit, the only prerequisite being the installation of the tool in the user community. There is the possibility of using and adapting task templates from an organizational database, or reusing old task specifications.

Simplicity, Genericity: the Task Manager in its present form is basically a very simple tool, and is usually very quickly understood by novice users as a distributed to-do list. It may be applied to any kind of collaborative activity, because it has not been tuned to any specific application domain.

Early designs and a first prototype of the Task Manager were evaluated in user workshops conducted in the spirit of the Scandinavian approach of user-centred design (Kyng and Greenbaum 1991). Our user organization is a company which manages a very large technical project, and our users at the workshops were managers, engineers, and support personnel; details on the organization may be found in (Grønbæk, Kyng and Mogensen 1992).

The main goal of the user involvement was to find out whether the Task Manager addressed the problems of distributed work management found in the user organization, and what might improve its design. The prospective users regarded the Task Manager as potentially useful for coordination within their organization, but thought *overview* and *efficiency* of the tool should be improved. This criticism has led to considerable enhancements of the Task Manager. We are sure that the planned real-world use will offer further occasions for improvements.

Finally, we found it interesting to compare our Task Manager with the check-list for a successful CSCW tool which recent analyses of cooperative work — computer-supported or not — came up with. Robinson (1993) claims that a tool should have a clear functionality to do a job and in addition should support the following features which we think are also present in the Task Manager.

Peripheral awareness: is offered by the task list interface which at a glance notifies of events in shared work the user should attend to.

Implicit communication: is supported via sharing documents and services attached to a task.

"Double level language", i.e. the complementary and mutually supportive use of implicit and explicit communication: is provided for by the conference of informal messages attached to a task.

Overview: of distributed work is one of the main features of the shared to-do list.

Multifunctionality: could be claimed as demonstrated in section 4, but still has to be proven in real use.

Our first assessments showed that distributed to-do lists can contribute to the management of distributed work. It also became clear that there still remains a lot to be done with regard to flexibility and interoperability of the tool. All of this has encouraged us to proceed with the further development of the Task Manager.

7 Conclusion, and: Where do we go from here

There is a still growing need for tools to manage distributed work. The coordination models and systems developed over the last years suffered from the *rigidity* of predefined procedures and imposed structures, and the *isolation* from informal communication and other forms of computer support.

With the Task Manager we have presented a simple, powerful, and generic tool for the management of distributed work that addresses these problems by its flexibility to adapt to a broad variety of collaborative work situations, and by the possibilities it offers to interoperate with other computer support.

The present prototype of the Task Manager will be the starting point for further evolutionary development. By putting the prototype to actual use we will further evaluate the tool and try to gradually add features that improve its usefulness as a coordination instrument. Additionally, we will address the problems connected with the mobile use of the Task Manager, i.e. intermittent disconnection from the base system, and further open it up by creating facilities to interoperate with other types of CSCW systems like distributed hypermedia and shared window conferencing.

Acknowledgements

We would like to thank all members of the EuroCoOp project team who participated in the design and implementation of the Task Manager, Andreas Bäcker, Ute Ehrlich, Pippa Hennessy, Karl-Heinz Klein, Ernst Lutz, Peter Seuffert, Alan Shepherd, and those who prepared and evaluated the user workshops, Kaj Grønbæk, Morten Kyng, Preben Mogensen and our users at Great Belt Link in Knudshoved.

References

Birman, K., Schiper, A., Stephenson, P. (1991): "Light weight causal and atomic group multicast," *ACM Transactions on Computer Systems*, vol. 9, no. 3, August 1991, pp. 272 - 314.

Grønbæk, K., Kyng, M., Mogensen, P. (1992): "CSCW challenges in large-scale technical projects: A case study," in J. Turner, R. Kraut (eds.): *CSCW '92*, Proc. Conf. on Comp.-Supp. Coop. Work, (Toronto, Oct. 31 - Nov. 4 1992), ACM, New York NY, pp. 338-345.

Hennessy, P., Kreifelts, Th., Ehrlich, U. (1992): "Distributed work management: activity coordination within the EuroCoOp project," *Computer Communications* vol. 15, no. 8, October 1992, pp. 477-488.

Kreifelts, Th., Hinrichs, E., Klein, K.-H., Seuffert, P., Woetzel, G. (1991): "Experiences with the DOMINO office procedure system," in L. Bannon, M. Robinson, K. Schmidt (eds.): *ECSCW '91*, Proc. 2nd European Conf. on Comp.-Supp. Coop. Work, Kluwer, Dordrecht, pp. 117 - 130.

Kyng, M., Greenbaum, J. (eds.) (1991): *Design at Work*, Lawrence Erlbaum, London.

Robinson, M. (1993): "Keyracks and computers: An introduction to 'common artefact' in Computer Supported Cooperative Work (CSCW)," *Wirtschaftsinformatik* vol. 35, no. 2, April 1993, pp. 157 - 166.

Rose, M. T., Onions, J. P., Robbins, C. J. (1991): "The ISO Development Environment: User's Manual," version 6.24, vols. 1 - 5.

Schmidt, K. (1991): "Riding a tiger, or computer supported cooperative work," in L. Bannon, M. Robinson, K. Schmidt (eds.): *ECSCW '91*, Proc. 2nd European Conf. on Comp.-Supp. Coop. Work, Kluwer, Dordrecht, pp. 1 - 16.

Steedman, D. (1990): *Abstract Syntax Notation One, The Tutorial Reference*, Technology Appraisals, Isleworth.