Transaction Support for Cooperative Work: An Overview of the TRANSCOOP Project[*]

Karl Aberer, Justus Klingemann, Thomas Tesch, Jürgen Wäsch, Erich Neuhold

GMD – German National Research Center for Information Technology
Dolivostraße 15, D-64293 Darmstadt, Germany
{aberer, klingem, tesch, waesch, neuhold}@darmstadt.gmd.de

Susan Even, Frans Faase, Peter Apers

Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
{seven, faase, apers}@cs.utwente.nl

Hannu Kaijanranta, Aarno Lehtola, Olli Pihlajamaa

VTT Information Technology
P.O. Box 1201, FIN-02044 VTT, Finland
{hka, all, ojp}@hemuli.tte.vtt.fi

Abstract

Cooperative work on shared information requires different kinds of computing system support to coordinate the work of multiple users, to establish mutual awareness among users, and to ensure the consistency of user results. These issues are currently tackled separately in various loosely related areas, such as workflow systems, groupware, and advanced transactional models. In the TRANSCOOP project, we have developed a transaction model and a specification language that provide a core functionality for information sharing in cooperative systems. The core functionality includes explicit work coordination facilities, which at the same time ensure the consistency of results.

The TRANSCOOP transaction model and specification language have evolved from a requirements analysis of various cooperative application scenarios. The transaction model has been implemented as an extension of an existing object-oriented database management system. The suitability of both the transaction model and the specification language to describe cooperative scenarios is being evaluated for a cooperative document authoring application.

1 Introduction

The availability of global information infrastructure has led to a rapid growth in opportunities to perform joint work in locally distributed environments and within virtual organisations. This requires the support of human interaction in cooperative working environments at a computing system level, and via conventional means, such as personal interaction, phone and mail. Applications where the globalisation of cooperative work is taking place include design applications (such as cooperative document authoring, CAD/CASE and design for manufacturing), real-time groupware (such as conferencing, shared whiteboards and joint editing), and business workflow management in administration and production.

Most of these applications are built upon some common information bases, which provide documents, design data or business data. Database systems managing this data thus need to support the typical modes of interaction of cooperating users with each other and with the computing system. These interactions involve aspects, such as multi-user cooperation on shared documents, support for long duration activities, or interactive user control. At the same time, basic consistency requirements need to be ensured. Ensuring consistency of data in multi-user environments is the classical problem of transaction management. However, in order to support cooperation, the classical paradigm of com-

[*]This document gives an overview of the work done in the ESPRIT III BRA project TRANSCOOP (EP8012); it has been prepared for distribution at the TRANSCOOP Workshop on Extending Data Management for Cooperative Work, held in Darmstadt, on June 6, 1997. This document is based on [AKT+96]. The TRANSCOOP project has been partially funded by the Commission of the European Communities. The partners in the project are GMD (Germany), University of Twente (The Netherlands), and VTT (Finland).

petition for resources needs to be replaced by the paradigm of semantically correct exchange and sharing of information.

The goal of the ESPRIT TRANSCOOP project has been the development of a cooperative transaction model and a corresponding specification language that are applicable to a wide spectrum of cooperative scenarios. The TRANSCOOP specification language COCOA allows the declarative specification of workflow-like cooperative scenarios [FEdBA97]. The TRANSCOOP cooperative transaction model COACT [RKT+95, WK96, KTW96b] provides the basic transactional support to ensure consistent management of shared data in cooperative applications.

The research activities in TRANSCOOP began with an analysis of different cooperative application scenarios, namely cooperative authoring [TW95], Design for Manufacturing [VFSE95] and workflow applications [JLP+95], and an analysis of existing approaches to support cooperative work. These analyses, together with the requirements derived from them, are presented in Section 2.

Section 3 describes the fundamentals of the TRANSCOOP cooperative transaction model COACT, and our assumptions about the structure of cooperative scenarios. The COACT model supports alternating periods of individual and joint work, and allows to exchange and share information consistently. To provide this flexibly, we take an operation-oriented view: the consistency of shared work results is determined, based on the semantics of the operations performed to obtain these results.

To specify cooperative scenarios in TRANSCOOP, we designed the specification language COCOA [FEdB96, FEdBA97], which allows the description of organisational and transactional aspects of a cooperative scenario, using declarative language constructs that extend a given database schema. The specification language is designed on top of the formal specification language LOTOS/TM [dBEV95, EFdB96], which in turn is based on the specification language LOTOS [BB89], and the TM database specification language and design tools, developed at the University of Twente [FB96, BBdB+96, vKSA+95, FvKS94, BdBZ93].

Within the TRANSCOOP project, the COACT transaction model has been implemented as an extension of the object-oriented database management system VODAK [GI95] (developed at GMD-IPSI). A TRANSCOOP demonstrator architecture has been implemented within the project to evaluate the transaction model and specification language for a particular application scenario, that of cooperative document authoring. Section 4 describes the components of the TRANSCOOP demonstrator system.

2 Requirements analysis and related approaches

2.1 Cooperative application domains investigated in TRANSCOOP

In the TRANSCOOP project, we investigated three kinds of cooperative application scenarios with respect to requirements for a cooperative transaction model and its specification language, namely *Cooperative Hypermedia Document Authoring* (CDA), *Design for Manufacturing* (DfM), and *Workflow applications* [TV95, VT95].

Cooperative Hypermedia Document Authoring [TW95]. CDA is characterised by multiple authors interactively working on shared hypermedia documents. Hypermedia document authoring can be considered as a design problem solving process [HF86], mainly characterised by the decomposition into smaller subproblems and their solution by interacting activities. An important characteristic of these processes is that the documents to be produced can be described only vaguely in advance. Authoring activities require a high degree of *flexibility* in choosing the next actions to end up with the aimed document.

Design for Manufacturing [VFSE95]. DfM can be seen as a variant of Concurrent Engineering. The scope of DfM is the engineering process of discrete complex industrial artifacts, usually separated into upstream processes (product design) and downstream processes (production realisation, including engineering, planning, and manufacturing) [SRN93]. The essential part of DfM is the early involvement of specialists from downstream processes in the upstream design process. Thus, the strengthening of the design process by *overlapping* design phases requires extensive cooperation and coordination facilities. In comparison to cooperative authoring, more detailed knowledge of the engineering processes in different phases, as well as of the sequence of processing is available.

Workflow applications [JLP+95]. Workflows are used to define complicated business processes, e.g., to accomplish the production of goods or services. A workflow consists of a collection of tasks that are partially ordered by control and data flow dependencies. Tasks are the basic units of work that are processed by one responsible actor. Thus, workflows focus mainly on the coordination of activities. Workflows may involve both automised/machine-based tasks, where a DBMS or other information systems are involved, and human-based tasks, where human beings are required to intervene and influence the flow of control, and indeed the cooperation.

The above cooperative application scenarios (CDA, DfM, and Workflow) can be classified by emphasising different aspects of the communication, collaboration and coordination properties of CSCW systems [EGR91]. In our investigation, we recognised the following characteristics (for details, refer to [TV95, VT95, TW95, VFSE95, JLP$^+$95]):

- Multiple concurrent users are involved in multiple activities to satisfy a common goal, or to produce a common product or artifact.
- Activities are processed interactively by humans and are usually of long duration.
- Cooperative work is characterised by alternating periods of individual and joint work.
- Exchange and sharing of persistent data between different users performing several activities is a basic feature.
- There is a need to ensure consistency both for the work of a single user, as well as for the cooperative effort.
- Between activities, there exist control flow dependencies that are derived from the application domain.
- It is likely that co-workers are geographically distributed and only partially connected, due to mobility.

The relevance of these characteristics varies for the three application domains. For example, in workflow scenarios, control flow dependencies are of higher importance than in the cooperative authoring domain. We can identify a spectrum, ranging from CDA, to DfM, to workflow applications, where the solutions become more fixed and prescribed, allowing the problem-solving process and termination conditions to be more deterministically described.

2.2 Requirements for cooperation support

A primary objective of our work in TRANSCOOP was to develop a cooperative transaction processing system that can support a broad spectrum of cooperative applications. Based on the above analysis, we observed the following requirements for such a system.

Relaxed atomicity. The rollback of the whole cooperative work process in the case of failure is generally not acceptable. It is required that a cooperative activity should be able to proceed (and eventually succeed) even if other parts of the cooperative process fail. A failure within one user's activity should not imply the rollback of another user's work in their joint effort.

Retraction of decisions. To support the interactive user control of activities, a cooperative transaction model has to provide services that allow to retract decisions taken by the cooperating users, for example by compensation. This allows, for instance, to explore several alternatives to solve a problem.

Support for exchange of results. The sharing of final, as well as of intermediate artifacts among co-workers is a prerequisite for most cooperative applications. A cooperative system should provide mechanisms to facilitate the exchange of tentative or partial results, while at the same time guaranteeing that no anomalies are introduced by the exchanges. The transaction model and its specification language should offer appropriate primitives for the semantically correct exchange of information between co-workers.

Support for private and shared data. To explore different solutions of the same problem, different co-workers should be able to work at the *same* time on the *same* data. To avoid interference from co-workers, the cooperative system should be able to manage alternative versions of objects. Upon a user's request, it should be possible to exchange versions, and to combine them into a commonly accepted version. Moreover, to support geographical distribution and mobility, the model should be able to deal with multiple copies of data.

Coordination of individual and joint work. In cooperative efforts different users work together. To achieve a common goal, facilities are needed to coordinate each user's work. In the more structured forms of cooperative scenario, such as workflow, coordination is coupled with the need to dynamically allocate tasks and assign co-workers to these tasks, based on the progress of the cooperation.

Scenario-dependent execution constraints. To describe the pre-planned parts of cooperative scenarios and the decomposition of the overall work into smaller subactivities, a cooperative system should allow the definition of execution constraints. Execution constraints should be able to describe the possible structure of a single user's activity as well as the overall cooperative work process. This includes the assignment of subactivities to co-workers and constraints on the occurrence of particular tasks and on their execution order within the overall cooperation process.

Of course, additional requirements are imposed on cooperative systems. For example, to support the mutual awareness of co-workers, notification mechanisms are needed. Additional communication facilities like e-mail or audio may be required for direct negotiation between co-workers. Such services received less attention in the TRANSCOOP project, because our goal was the synchronisation of cooperative access to persistent data. Nevertheless, a cooperative transaction processing system satisfying the above requirements can provide an application-independent nucleus on which to build cooperative systems.

2.3 Related approaches

The issue of synchronisation of multiple cooperating users is treated in several fields, including groupware, workflow systems, and advanced transaction models.

Groupware. Most groupware systems [EGR91] synchronise cooperative access to shared data in a more or less *ad hoc* manner. Concurrency control in most cooperative hypertext systems is based on mechanisms like explicit user-controlled locking of objects, different lock modes, extended lock semantics, and notifications [WL93, GS87]. Some systems use floor passing protocols [GS87] to synchronise concurrent operations on shared data, thereby limiting the availability of data. Some systems do not provide any concurrency control at all, but rely on social protocols [EGR91]. Other approaches in the CSCW area (e.g., [EG89]) are only applicable to real-time groupware systems, such as shared whiteboards and synchronous group editors. Most of these systems are based on replication of data and use multicast protocols like ISIS [BC91, BSS91] for synchronisation purposes. Real-time groupware systems do not address the issues of persistency of data and recovery to ensure fault-tolerant processing.

Workflow systems. Workflow management is gaining popularity, although the current generation of workflow management systems (WFMS) has several limitations [GHS95, JLP+95]. Most of these arise because the purely process-centric approach of WFMS neglects data-centric issues. This results in the lack of support for correctness and data consistency (in the case of concurrent workflow tasks), and insufficient recovery mechanisms. Most of the commercial WFMS concentrate on relatively static workflows. Some of the applications we studied in the TRANSCOOP project have a somewhat statically defined activity structure at a high-level, but at a lower level, the order of executions of the constituent activities is dynamically determined during scenario execution by the participating users. Current WFMS do not have adequate modelling and execution support for this.

Advanced transaction models. Transaction models guarantee fault tolerance and synchronise concurrent access to shared persistent data. To support cooperative applications, several advanced transaction models have been proposed in the recent years. For an overview of these models we refer the reader to [Elm92, Hsu93, Kai95].

A general approach to supporting cooperation is to divide the database into public and private areas [KW84, LP83]. Objects are then copied from the public database by check-out into private areas. When a transaction is finished, the modified objects are checked-in into the public database. Check-out models often appear in tandem with versions and configurations [Kat90]. CAD transactions [BKK85, KSUW85] enhance the basic check-out model by introducing a hierarchy of public, semi-public, and private databases. For CASE applications, several extensions of the basic checkout model have been developed, which take advantage of the opportunity for generic software consistency checking [Hon88, KPS89], but these models are not generally applicable to other domains.

The split/join transaction model [PKH88, KP92] supports the dynamic restructuring of ongoing transactions. A split-operation allows to split a running transaction into two new (serial or independent) transactions while a join-operations allows to incorporate two transactions into a new transaction. These mechanisms enable a cooperative behaviour by exchanging parts of transactions between concurrent users.

Transactional workflow approaches [RS95] usually include specification languages to express various execution constraints for a set of tasks. This can be done either by supporting a script language (as in the ConTract model [WR92]), by a declarative specification of the execution structure in terms of externally visible execution states [ARSS93], or by ECA rules [DHL90]. Cooperation is characterised in these models by passing results between workflow tasks in a predefined manner.

3 The TRANSCOOP Model

As described in Section 2, cooperative work has several dimensions, and the requirements for a cooperative system largely vary depending on the application. In TRANSCOOP, it was our aim to satisfy these requirements by introducing a set of complementary mechanisms, which can be used by the application designer to tailor the system for specific needs. In the following, we describe the building blocks of the TRANSCOOP transaction model and specification language, together with the aspects of cooperative work that they support.

3.1 Workspaces and exchange facilities

The *Cooperative Activity Model* (COACT) comprises the core of the TRANSCOOP transaction model [RKT+95, WK96, KTW96b]. The model builds on the observation stated in Section 2 that cooperative work is characterised by alternating periods of individual and joint work [TV95, TW95, Kai95]. During individual work periods, users try out alternative problem solutions while co-workers may work simultaneously on the same subject. Access to and use of shared data should neither block other users, nor should it affect co-workers unintendedly. During joint work, co-workers should be able to exchange information and to share final as well as intermediate results. Moreover, dynamic subgrouping of co-workers should be possible.

In COACT, we assign a *private workspace* to every user who takes part in a cooperative activity. By default, the private workspaces of the co-workers are isolated from each

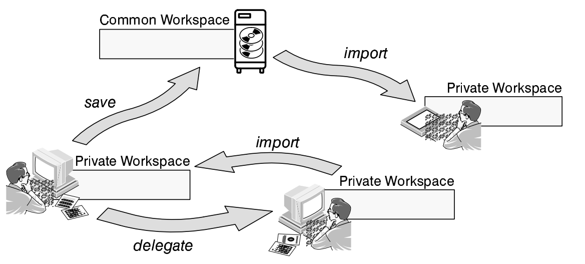

Figure 1: Workspaces and exchange facilities in COACT.

other. Additionally, there is also a *common workspace* for each cooperative activity. This workspace is isolated from the private workspaces; it is not assigned to any single user in the cooperative activity. The common workspace contains the data items available when a cooperative activity is *started*, and it will contain the results of the cooperative activity when the activity is *committed*. Figure 1 gives an overview of these constituents of the COACT model.

To achieve isolation of workspaces, we (conceptually) copy the data items initially contained in the common workspace to all private workspaces. From these copies, the users can create (throughout the working process) their own private versions of data items, which can be independently manipulated. Hence, the modifications to data items done by different co-workers do not interfere. For each workspace, a log of the modifications is kept in a *workspace history*.

Each cooperative activity is described by (1) a set of operations that can be invoked by a user in the user's private workspace, and (2) a set of type-specific merging rules that exploit the semantics of operations to guide the process of information exchange (*history merging*) [WK96, KTW96b].

Operations are the smallest units of work within a cooperative activity. An operation is considered to be atomic and transfers a consistent workspace state to another consistent state. The COACT model assumes an environment where the sequence of operations executed in a workspace is composed interactively, by the user at run-time (as in the hypermedia authoring system SEPIA [SHH$^+$92, WA95, BWAH96]). A user selects a next operation from the predefined operation set associated with the cooperative activity. The actual input parameters to the operation are given explicitly by the user.

Information exchange among workspaces in COACT is based on the exchange of operations instead of data. This is an explicit act that is initiated by an actor by invoking one of COACT's *exchange operations*. The exchange operations are generic meta-operations of the COACT model (like starting, aborting, or committing of a cooperative activity); they are based on the paradigm of merging workspace histories. The COACT model provides two different options for exchanging information:

1. Co-workers can directly exchange operations between their private workspaces by means of *import* and *delegate* operations. The import operation is used by a co-worker to incorporate operations executed in the scope of another workspace into the local workspace. The importing user is responsible for resolving conflicts that may occur during the merge. The delegate operation is used to pass on a set of operations to a co-worker who is then responsible for merging them into the destination workspace.

2. Co-workers can exchange operations through the common workspace by means of *save* and *import* operations. A user can invoke the save operation to incorporate operations of the user's private workspace into the common workspace, thus making (partial) results public to all co-workers. The user who invokes the save operation is responsible for the resolution of conflicts. Other co-workers can then retrieve the saved information using the import operation described above.

The COACT history merge mechanism ensures that only consistent parts of workspaces are exchanged. Consistent units of work are identified by examining the *backward commutativity* relation [Wei88, LMWF94] between operations contained in a workspace history. The incorporation of operations into a workspace is then realised by the *re-execution* of the operations in the destination workspace. In this way, the effects of these operations are reflected in the private versions of the data items in the destination workspace.

The semantic correctness of the exchange of operations is guaranteed by ensuring that the re-execution of an operation has an equivalent “view” on the history, in both the destination workspace and the source workspace. Hence, the behaviour of a re-executed operation in terms of output results is indistinguishable from its initial execution. We use the *forward commutativity* relation [Wei88, LMWF94] to check this. If the merge process cannot be performed without violating the semantical correctness, the merge algorithm identifies different consistent sets of operations; one of these solutions can then be used in the merge instead. The TRANSCOOP run-time system offers facilities to the user to support this selection process. Merging may result in reverting previous decisions. This is done by compensation in COACT. In [KTW96b], we have shown that this approach produces correct merged histories.

If an operation has been successfully incorporated into another workspace, it is conceptually the same operation that is present in *more than one workspace*. The presence of identical operations in several workspaces enables us to establish a close cooperation between co-workers, because conflicts between operations that are present in both workspace histories can be ignored. Conceptually, the co-workers owning the workspaces have already agreed on the results of these “shared” operations.

Those parts of a cooperative activity that are reflected in the common workspace after its completion (commit) are considered as its final result. It is assumed that all users integrate their relevant contributions into the common workspace to produce a single result of the cooperative activity. The merge mechanism and its properties are discussed in detail in [KTW96b, WK96].

Facilities to define data operations, data exchange operations, and the forward and backward commutativity relations required by the COACT transaction model are available in the COCOA specification language for cooperative scenarios [FEdB96, FEdBA97]. COCOA extends the object-oriented database schema specification language TM [BBdB+96]. The data exchange mechanism of COCOA allows the specification of queries over the operations that exist in a workspace history to select operations for exchange (e.g., import or delegation).

3.2 Organising the work process

The merge mechanism of COACT enables the participants in a cooperative activity to work concurrently on the same data items without blocking each other, while at the same time ensuring the consistency of their results. This is a basic requirement for all cooperative applications and may even be sufficient for the support of creative work that is performed in a more or less *ad hoc* fashion, such as cooperative document authoring. Other cooperative applications such as workflow require additional mechanisms to *coordinate* the work process. These mechanisms must be capable of ensuring, for example, that certain operations are performed in a specific order, or that certain operations are indeed performed. Such coordination mechanisms are available within the TRANSCOOP model at two levels: at the workspace level, and at the cooperative scenario level.

With regard to single workspaces, the application designer can specify a set of *execution rules* in COCOA. These rules pose workflow-like restrictions on the order and existence of operations in a workspace history, and define termination states. Execution rules have to be enforced for each private workspace and the common workspace separately. We use a language-based specification mechanism in COCOA for the execution rules. The application designer specifies certain grammars, and the allowed sequences of operations are equal to the words in the generated language. To specify this language, multiple grammars can be used. Their combination results in the desired restrictions on sequences of operations that are allowed by the cooperative scenario. In contrast to [Ska89, NRZ92], our mechanism avoids dead ends caused by interdependencies of different grammars. For details see [KTW+96c, FEdB96].

To structure the overall work process, and to specify restrictions that must be obeyed across all workspaces, COCOA provides a step definition mechanism [FEdB96]. The step mechanism controls whether a user is allowed to execute an operation at a particular point in the scenario. This is performed by explicitly *enabling* the allowed operations. The set of operations that are controlled by the step mechanism include data operations and exchange operations, as well as communication operations to initiate sub-steps, and/or to terminate a step.

A step governs the operation enabling in multiple workspaces, but it is up to the users to decide whether they want to execute the enabled operations, and in what order. In contrast to execution rules, there are no restrictions on the order or existence of operations executed *within* a step. The enabling mechanism is complemented by the possibility to specify *user roles*. Hence, the permission to execute an operation can be restricted to users filling a certain role. Steps can be combined in various ways to form the organisational structure of the work process. The constructs to group steps range from sequential and repetitive execution of steps, to nested and parallel steps. For more details on scenario organisation facilities in COCOA, see [FEdB96].

4 The TRANSCOOP System

To demonstrate the applicability of the results of the TRANSCOOP project, an important goal was to implement a prototype system that realises the most important concepts introduced in the model. Due to time limitations, the execution rule enforcement mechanism has not been implemented. The TRANSCOOP reference architecture [dBLP+95] identifies two fundamental components: the *specification environment*, which provides a means for the specification and verification of a cooperative scenario, and the *runtime environment*, which offers support for the execution of a cooperative scenario. These are described in the following sections.

4.1 The specification environment

The components of the TRANSCOOP specification environment are intended to help the specifier of a cooperative application work with the COCOA language in the early (conceptual), as well as the late (testing) phases of the application design [EFPdB96]. The TRANSCOOP specification environment includes a graphical specification editor, a static analysis tool, a dynamic analysis tool, and compilers to the run-time environment. The static analysis tool includes a parser/type-checker, which also performs well-formedness checks on the steps and transitions. The dynamic analysis tool offers simulation and visualisation of the organisational aspects of the cooperative scenario using the TM Abstract Machine. The compilers generate the input for the cooperation manager and the cooperative transaction manager of the run-time environment.

4.2 The run-time environment

The TRANSCOOP runtime environment has been implemented as an extension of the object-oriented DBMS VODAK, developed at GMD-IPSI [GI95]. In the implementation architecture, all cooperation facilities are supported as DBMS services. The design of the TRANSCOOP runtime environment required an extension of the traditional, centralised OODBMS architecture (e.g., as used for VODAK) in order to meet the architectural and conceptual requirements posed by the TRANSCOOP cooperative transaction manager [KTW96a]. The following issues were addressed:

- **Workspaces.** The desired workspace functionality requires the maintenance of multiple private versions of an object, instead of a single shared version. To meet this requirement, the object management strategy (initially based on a single, centralised shared object buffer) was replaced by multiple private object buffers, which can be distributed across the network.

- **History maintenance.** The merge algorithm on which the TRANSCOOP exchange facilities are based uses the workspace histories of different workspaces. To provide efficient access to *all* workspace histories in an ongoing scenario, the histories are centrally stored in the TRANSCOOP system.

  When an operation is invoked that manipulates the local workspace state, the corresponding history entry is stored persistently in a dedicated *status database*, which maintains the current state of the scenario execution. This design decision requires permanent availability of the database server, but, in the case of failure on the client side, it allows the recovery of the client's workspace state. An alternative design with local storage of histories would decrease the client's dependence on the database server. This would be more suitable for the purpose of mobility.

- **Scenario meta-data.** For the execution of cooperative scenarios, the VODAK database schemas are enriched by a cooperative scenario specification. In particular, this captures (1) information about which schema methods are available at the application interface, (2) the structure of the overall scenario execution (steps), (3) information about how certain method executions can be compensated, and (4) predicates describing how to evaluate backward and forward commutativity at runtime. All of this additional information is stored in the enhanced VODAK data dictionary.

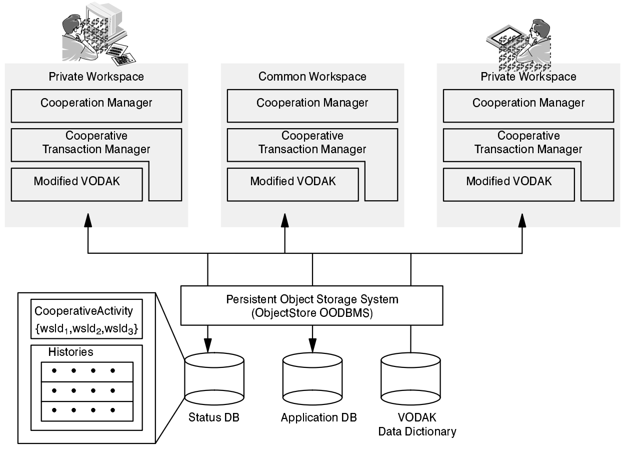

Figure 2: Run-time architecture of the system.

Figure 2 illustrates the runtime architecture of the TRANSCOOP system. The workspaces are realised by a modified VODAK system, which includes *private object buffers*, the *cooperative transaction manager* (containing the merge functionality), and the *cooperation manager* (governing the execution of steps).

An important part of the runtime system design is the Status DB, which maintains the histories of the participating users and the runtime structure of the step specification; it also provides general administrative information about the state of the scenario, such as the current participants in the scenario, and the operations enabled by the current steps.

The cooperative transaction manager offers functions to preview the progress of co-workers by means of querying their histories in the Status DB, and it provides operations for both the calculation of possible merge alternatives, and the enforcement of a specific merge.

Users can join an ongoing scenario execution as well as leave the scenario before it is finished. When the current step allows the scenario to terminate, the results in the common workspace represent the commonly agreed upon result. The final history is then applied to the Application DB.

4.3 The TRANSCOOP demonstrator scenario

The selected application for the TRANSCOOP demonstrator system [dBLP+96] is the cooperative hypermedia authoring system SEPIA [SHH+92], developed at GMD-IPSI.

Because we treat the authoring process from a database point of view, we assume a scenario in which multiple authors are manipulating a collection of shared hypermedia documents that are stored in a database management system. The SEPIA hierarchical hypertext document structure and the corresponding operations for the creation and manipulation of SEPIA document structures are modelled explicitly by means of the VODAK Manipulation Language (VML) in the database schema [WA95, BWAH96].

The TRANSCOOP demonstrator system provides information about the current participants and their work progress to establish group awareness. For example, when a delegation is performed, the delegatee is notified in order to control the integration of the respective piece of work into his or her workspace. Further notifications are caused if co-authors join or leave the cooperative activity.

The exchange facilities of the COACT model support an *ad hoc* working style of groups with no predefined coordination. Conflicts that may occur between the individually carried out problem solutions are semi-automatically resolved within the merge procedure. The cooperative transaction manager offers different consistent alternatives in case of a conflict. The controlling user then selects one of the offered solutions using a graphical user interface.

5 Conclusions

Today's workflow systems do not support scenarios with spontaneous cooperation requirements that cannot be prescribed *a priori* in a specification. In contrast, current groupware approaches support an *ad hoc* working style, yet give, from a database perspective, no satisfying execution guarantees. The TRANSCOOP system addresses both criteria: the growing need for models supporting highly dynamic forms of cooperative work, as well as the need for transactional correctness criteria.

The nucleus of the exchange facilities in the COACT model is the history merging approach. The flexibility of the approach is mainly achieved by its ability to determine consistent units of work dynamically, in terms of performed operations, and its consideration of operation semantics for resolving conflicts.

A question that requires further attention is the specification of commutativity relations. Specification tools that compute commutativity information automatically from a formal specification of data operations will improve the practical applicability of the TRANSCOOP approach. Within the TRANSCOOP project, we have made preliminary investigations in this direction. This investigation involves mapping a TM schema to higher order logic (HOL) [Spe95]. Commutativity analysis requirements are formulated as proof goals in HOL and given, along with the HOL representations of the data operations, to the Isabelle theorem prover for proof assistance [Pau90, Pau94].

The TRANSCOOP approach fits various application areas and provides directions for solving open research problems in related fields like mobile wireless computing [KTW97] or versioning.

Acknowledgements

The authors would like to thank the following former members of the TRANSCOOP team, who have left the institutions involved in the project: Rolf de By (formerly at the University of Twente, now at ITC Enschede, The Netherlands), Wolfgang Klas (formerly at GMD, now at Universität Ulm, Germany), Jari Veijalainen (formerly at VTT, now at the University of Jyväskylä, Finland), and Peter Verkoulen (formerly at the University of Twente, now at Origin IT Services, Eindhoven, The Netherlands).

References

[AKT+96] K. Aberer, J. Klingemann, T. Tesch, J. Wäsch, and E. J. Neuhold. Transactional Models Supporting Cooperative work – The TRANSCOOP Experiences. In *Proceedings of the International Symposium on Cooperative Database Systems for Advanced Applications (CODAS)*, pages 467–476, December 1996. Kyoto, Japan.

[ARSS93] P. C. Attie, M. Rusinkiewicz, A. Sheth, and M. P. Singh. Specifying and enforcing intertask dependencies. In *Proc. of the 19th Int. Conference on Very Large Databases*, pages 134–145, Dublin, Ireland, August 1993.

[BB89] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1989.

[BBdB+96] René Bal, Herman Balsters, Rolf A. de By, Alexander Bosschaart, Jan Flokstra, Maurice van Keulen, Jacek Skowronek, and Bart Termorshuizen. The TM Manual, version 2.0, revision f. Technical Report IMPRESS/UT-TECH-T79-001-R2, Universiteit Twente, The Netherlands, February 1996.

[BC91] K.P. Birman and R. Cooper. The ISIS project: Real experience with a fault-tolerant programming system. *ACM Operating System Review*, 21(2):103–107, 1991.

[BdBZ93] H. Balsters, R. A. de By, and R. Zicari. Typed sets as a basis for object-oriented database schemas. In Oscar M. Nierstrasz, editor, *Proceedings of the Seventh European Conference on Object-Oriented Programming*, volume 707 of *LNCS*, pages 161–184, Kaiserslautern, Germany, 1993. Springer.

[BKK85] F. Bancilhon, W. Kim, and H. Korth. A model of CAD transactions. In *Proc. of the 11th Int. Conference on Very Large Databases*, pages 25–33, Stockholm, Sweden, August 1985.

[BSS91] K.P. Birman, A. Schiper, and P. Stephenson. Leightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, 1991.

[BWAH96] A. Bapat, J. Wäsch, K. Aberer, and J.M. Haake. HyperStorM: An extensible object-oriented hypermedia engine. In *Proceedings of the Seventh ACM Conference on Hypertext (HYPERTEXT'96)*, pages 203–214, March 16-20 1996. Washington, D.C.

[dBEV95] R. A. de By, S. J. Even, and P. A. C. Verkoulen. Functionally specified distributed transactions in co-operative scenarios. In *Proceedings of RIDE-DOMS*, pages 116–121, Taipei, Taiwan, March 1995.

[dBLP$^+$95] R. de By, A. Lehtola, O. Pihlajamaa, J. Veijalainen, and J. Wäsch. A reference architecture for cooperative transaction processing systems. VTT Research Notes 1694, VTT Technical Research Centre of Finland, 1995.

[dBLP$^+$96] R. de By, A. Lehtola, O. Pihlajamaa, J. Veijalainen, and J. Wäsch. Deliverable III.2: Specification of the Demonstrator. Report TC/REP/VTT/D3-2/960425, Esprit Project No. 8012, 1996.

[DHL90] U. Dayal, M. Hsu, and R. Ladin. Organizing long–running activities with triggers and transactions. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 204–214, Atlantic City, NJ, USA, May 1990.

[EFdB96] Susan J. Even, Frans J. Faase, and Rolf A. de By. Language features for cooperation in an object-oriented database environment. *International Journal of Cooperative Information Systems, Special Issue on Formal Methods*, 5(4):469–500, December 1996.

[EFPdB96] Susan J. Even, Frans J. Faase, Olli Pihlajamaa, and Rolf A. de By. Deliverable IV.4: Design of the TransCoop Specification Environment. Report TC/REP/UT/D4-4/032, Esprit Project No. 8012, 1996.

[EG89] C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 399–407. MCC, Austin, Texas, May 1989. Portland, Oregon.

[EGR91] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, January 1991.

[Elm92] A. K. Elmagarmid, editor. *Database Transaction Models for Advanced Applications.* ACM Press. Morgan Kaufmann Publishers, Inc., 1992.

[FB96] Jan Flokstra and Reinier Boon. The TM Abstract Machine (TAM). Internal working document, Universiteit Twente, The Netherlands, February 1996.

[FEdB96] Frans J. Faase, Susan J. Even, and Rolf A. de By. Deliverable IV.3: An introduction to CoCoA. Report TC/REP/UT/D4-3/033, Esprit Project No. 8012, 1996.

[FEdBA97] Frans J. Faase, Susan J. Even, Rolf A. de By, and Peter M. G. Apers. Integrating organisational and transactional aspects of cooperative activities. Accepted for publication, *Workshop on Database Programming Languages (DBPL)*, 1997.

[FvKS94] Jan Flokstra, Maurice van Keulen, and Jacek Skowronek. The IMPRESS DDT: A database design toolbox based on a formal specification language. In *Proc. of ACM-SIGMOD '94*, Minneapolis, Minnesota, 1994.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[GI95] GMD-IPSI. VODAK V4.0 User Manual. Arbeitspapiere der GMD 910, Technical Report, GMD, April 1995.

[GS87] I. Greif and S. Sarin. Data sharing in group work. *ACM Transactions on Office Information Systems*, 5(2):187–211, April 1987.

[HF86] J.R. Hayes and L. Flowers. Writing research and the writer. *American Psychologist*, 41(10):1106–1113, 1986.

[Hon88] M. Honda. Support for parallel development in the Sun network software environment. In *Proc. of the second Int. Workshop on Computer-Aided Software Engineering*, pages 5–5 – 5–7, Cambridge, Massachusetts, USA, July 1988.

[Hsu93] M. Hsu, editor. *Data Engineering Bulletin, Special Issue on Workflow and Extended Transaction Systems.* IEEE Press, 1993. Vol. 16, No. 2.

[JLP$^+$95] J. Juopperi, A. Lehtola, O. Pihlajamaa, A. Sladek, and J. Veijalainen. Usability of some workflow products in an inter-organizational setting. In *Proc. of IFIP WG8.1 Working Conference on Information Systems for Decentralized Organizations*, Trondheim, Norway, August 1995.

[Kai95] G. E. Kaiser. Cooperative transactions for multiuser environments. In Kim [Kim95], chapter 20, pages 409–433.

[Kat90] R. H. Katz. Towards a unified framework for version modelling in engineering databases. *ACM Computing Surveys*, 22(4), 1990.

[Kim95] W. Kim, editor. *Modern Database Systems: The Object Model, Interoperability, and beyond.* Addison-Wesley Publishing Company, 1995.

[KP92] G. E. Kaiser and C. Pu. Dynamic restructuring of transactions. In Elmagarmid [Elm92], chapter 8, pages 265–295.

[KPS89] G. E. Kaiser, D. E. Perry, and W. M. Schell. Infuse: Fusing integration test management with change management. In *Proc. of the 13th IEEE Computer Software and Applications Conference*, pages 552–558, Orlando, Florida, USA, September 1989.

[KSUW85] P. Klahold, G. Schlageter, R. Unland, and W. Wilkes. A transaction model supporting complex applications in integrated information systems. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 388–401, Austin, Texas, USA, May 1985.

[KTW96a] J. Klingemann, T. Tesch, and J. Wäsch. Deliverable V.3: Design of the TransCoop Cooperative Transaction Manager. Report TC/REP/GMD/D5-3/512, Esprit Project No. 8012, 1996.

[KTW96b] J. Klingemann, T. Tesch, and J. Wäsch. Semantics-based transaction management for cooperative applications. In *Proc. of the Int. Workshop on Advanced Transaction Models and Architectures*, Goa, India, August 31 – September 2 1996.

[KTW$^+$96c] J. Klingemann, T. Tesch, J. Wäsch, J. Puustjärvi, and J. Veijalainen. Deliverable V.2: Definition of the TransCoop Cooperative Transaction Model. Report TC/REP/GMD/D5-2/511, Esprit Project No. 8012, 1996.

[KTW97] J. Klingemann, T. Tesch, and J. Wäsch. Enabling cooperation among disconnected mobile users. In *Proc. of the second IFCIS Int. Conference on Cooperative Information Systems (CoopIS)*, June 1997.

[KW84] R. Katz and S. Weiss. Design transaction management. In *Proceedings of the 19th Design Automation Conference*, June 1984.

[LMWF94] N. Lynch, M. Merrit, W. Weihl, and A. Fekete. *Atomic Transactions.* Morgan Kaufmann Publishers, Inc., 1994.

[LP83] R. Lorie and W. Plouffe. Complex objects and their use in design transactions. In *Proceedings on Database for Engineering Applications*, pages 115–121. ACM, May 1983.

[NRZ92] M. H. Nodine, S. Ramaswamy, and S. B. Zdonik. A cooperative transaction model for design databases. In Elmagarmid [Elm92], chapter 3, pages 53–85.

[Pau90] Lawrence C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[Pau94] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *LNCS*. Springer-Verlag, 1994.

[PKH88] C. Pu, G. E. Kaiser, and N. Hutchinson. Split–transactions for open–ended activities. In *Proc. of the 14th Int. Conference on Very Large Databases*, pages 26–37, Los Angeles, California, USA, August 1988.

[RKT$^+$95] M. Rusinkiewicz, W. Klas, T. Tesch, J. Wäsch, and P. Muth. Towards a cooperative transaction model: The cooperative activity model. In *Proc. of the 21st Int. Conference on Very Large Databases*, pages 194–205, September 1995. Zurich, Switzerland.

[RS95] M. Rusinkiewicz and A. Sheth. Specification and execution of transactional workflows. In Kim [Kim95], chapter 29, pages 592–620.

[SHH$^+$92] N. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt, and M. Thüring. SEPIA: A cooperative hypermedia authoring environment. In *Proc. of the fourth ACM Conference on Hypertext*, pages 11–22, 1992. Milano, Italy, Nov. 30 – Dec. 4.

[Ska89] A.H. Skarra. Concurrency control for cooperating transactions in an object-oriented database. *ACM SIGPLAN Notices*, 24(4):145–147, April 1989.

[Spe95] David Spelt. A Proof Tool for TM. M.Sc. Thesis, Universiteit Twente, July 1995.

[SRN93] A. Storr, U. Rembold, and B.O. Nnaji. *Computer Integrated Manufacturing and Engineering.* Addison-Wesley Publishing Company, 1993.

[TV95] T. Tesch and P. Verkoulen. Deliverable II.2: Requirements for the TransCoop Transaction Model. Report TC/REP/GMD/D2-2/207, Esprit Project No. 8012, 1995.

[TW95] T. Tesch and J. Wäsch. Transaction support for cooperative hypermedia document authoring: A study on requirements. In *Proc. of 8th ERCIM Database Research Group Workshop on Database Issues and Infrastructure in Cooperative Information Systems*, pages 31–42, Trondheim, Norway, August 1995.

[VFSE95] P. A. C. Verkoulen, F. J. Faase, A. W. Selders, and P. J. J. Oude Egberink. Requirements for an advanced database transaction model to support Design for Manufacturing. In *Proceedings of the Flexible Automation and Intelligent Manufacturing Conference*, pages 102–113. Begell House, Inc. New York - Wallingtord (U.K.), June 1995. Stuttgart, Germany.

[vKSA$^+$95] M. van Keulen, J. Skowronek, P.M.G. Apers, H. Balsters, H.M. Blanken, R.A. de By, and J. Flokstra. A framework for representation, validation and implementation of database application semantics. In *Proc. of the sixth IFIP Conference on Database Semantics*, May 30 – June 2 1995. Atlanta, Georgia.

[VT95] P. Verkoulen and T. Tesch. Deliverable II.1: Requirements for the TransCoop Specification Language. Report TC/REP/UT/D2-1/014, Esprit Project No. 8012, 1995.

[WA95] J. Wäsch and K. Aberer. Flexible design and efficient implementation of a hypermedia document database system by tailoring semantic relationships. In *Proc. of the sixth IFIP Conference on Database Semantics*, May 30 – June 2 1995. Atlanta, Georgia.

[Wei88] W. E. Weihl. Commutativity-based concurrency control for abstract data types. *IEEE Transactions on Computers*, 37(12):1488–1505, 1988.

[WK96] J. Wäsch and W. Klas. History merging as a mechanism for concurrency control in cooperative environments. In *Proceedings of RIDE-Interoperability of Nontraditional Database Systems*, pages 76–85, New Orleans, USA, February 1996.

[WL93] U. K. Wiil and J. J. Leggett. Concurrency control in collaborative hypertext systems. In *Proc. of the fifth ACM Conference on Hypertext*, pages 14–18, November 1993. Seattle, Washington.

[WR92] H. Wächter and A. Reuter. The ConTract model. In Elmagarmid [Elm92], chapter 7, pages 219–264.