Frank Leymann, Hans-Jörg Schek,
Gottfried Vossen (editors):

Transactional Workflows

Dagstuhl-Seminar-Report; 152
15.07.-19.07.95 (9629)

ISSN 0940-1121

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloss Dagstuhl
Universität des Saarlandes
Postfach 15 11 50
D-66041 Saarbrücken, Germany
Tel.: +49 -681 - 302 4396
Fax: +49 -681 - 302 4397
e-mail: office@dag.uni-sb.de
url: http://www.dag.uni-sb.de

Transactional Workflows

Organizers: F. Leymann, H.-J. Schek, G. Vossen

Dagstuhl Seminar 9629

1 The Subject

1.1 Transactional Workflows: An Introduction

by Gottfried Vossen

Workflows are activities involving the coordinated execution of multiple tasks performed by a number of different processing entities [6, 10]. In this context, *tasks* represent work to be done and can be specified in a variety of ways, including textual descriptions (e.g., in e-mail or a file), forms, messages, or computer programs. *Processing entities* which can perform tasks may be humans or software systems, e.g., mailers, application programs, database management systems. Workflow executions are controlled and coordinated by software systems called *workflow management systems* (WFMS, [3, 12]). The modeling, specification, simulation, verification, and optimization of extensible, adaptable, and scalable workflows, and the design and implementation of WFMS has recently gained considerable interest, due to the recognition that information system applications nowadays are heterogeneous, distributed, and need to integrate hosts of different technologies, legacy systems, and existing software, and that a computer-aided coordination of users of such systems who are involved in the execution of some enterprise process is of increasing importance [11].

Workflow management combines influences from a variety of disciplines, including cooperative information systems, computer-supported cooperative work (CSCW), groupware systems, or active databases. On the other hand, workflows can be considered as a generalization of multidatabase transactions [5], i.e., transactions accessing multiple databases which are locally autonomous. Indeed, multidatabase transactions can be seen as special workflows in which structuring, isolation, and atomicity properties are determined by the underlying transaction model. However, for workflows that are executed in an environment of heterogeneous processing entities transaction models which were developed for distributed database systems are rarely sufficient. For example, transaction models typically provide a predefined set of properties which may or may not be required by the semantics of a workflow; conversely, processing entities involved in workflow execution may not provide support for facilities required by a specific transaction model, the reason being that transaction models are tailored towards processing entities which are database systems [1, 4, 10].

Nevertheless, there are good reasons for assuming that what has been learned over the years in the context of database transaction models and processing can fruitfully be exploited for workflow management. For example,

- a variety of advanced transaction models has been proposed in recent years which aims at reflecting issues like modular software construction, user control over executions, modeling of complex or long-lived activities, cooperation, interactiveness, or system federations [2, 7, 8];
- for executing multiple transactions concurrently, a number of correctness criteria has been proposed which go considerable beyond traditional serializability [2];

- a great body of experience now exists in designing and implementing scheduling devices for database systems and transaction processing (TP) monitors [9].

When enterprises run their business based on WFMS, these systems must have the same industrial strength and robustness that, e.g., database management systems have today. Moreover, advanced processing paradigms like "disconnected users" are to be supported in such environments. Thus, architectural lessons learned by the database community or the TP monitor community can contribute to WFMS architectures, but several new issues must be incorporated into such architectural frameworks.

Another relevant area of application for WFMS is Business Process Reengineering (BPR, [12]). One of its major goals is to allow an enterprise to react fast in dynamic environments. This requires the extraction of flow information (control, data, organization) from application systems, to allow for the required flexibility of the software. The resulting application structure also stimulates reuse, and enhances productivity of application developers (e.g., composing applications from parts). In this context, the relationship of the workflow paradigm to the object paradigm becomes relevant.

The goal of this seminar was to bring together (1) researchers from the areas of databases and information systems, and (2) developers as well as users of WFMS, in order to get to an in-depth understanding of the issues, and to discuss demanding questions and open problems in detail. These issues include

- the distinctive features of workflow management, as opposed to, for example, CSCW, groupware, or conferencing systems,
- a taxonomy of different workflow definitions, both from an informal and from a formal point of view,
- the impact of advanced transaction models on workflow models,
- the relevance of the ACID properties,
- suitable correctness criteria for single and for concurrent workflow executions,
- architectures for WFMS and their relationship to recent standardization efforts and middleware components in distributed computing, e.g., CORBA or DCE.

Since there are many different notions of workflow, ranging from a loose description of a business process to rigorous definitions given in several theoretical papers, it is important to classify different definitions along with their applicability to a solution of various practical and practically important theoretical issues of the WFMS systems. Such a taxonomy can then be made fit with different notions of extended transactions models, their correctness criteria, and their processing environments. As was said above, advanced transaction models typically capture some aspects of workflows, but not all; nevertheless, models like multi-level transactions or ConTracts, which relax atomicity and isolation commonly associated with a transaction, appear applicable in specific workflow contexts. Moreover, TP monitors can ensure transaction atomicity in a homogeneous distributed environment. On the other hand, currently available WFMS allow

an execution of a workflow in a heterogeneous distributed environment but without guaranteeing something like transactional ACID properties.

Regarding correctness, more research is needed on the question of how to extend classical correctness notions for concurrent transaction executions to workflow management in a simple, yet formally precise and technically adequate way. This seems to be a non-trivial task, due, for example, to the fact that workflows typically have to maintain global invariants, while their constituent transactions need to preserve local constraints. Additionally, workflow correctness requirements are often hidden in their specification, as part of the "knowledge" about the underlying application and its dynamics. Another aspect is that system architectures for WFMS need to enforce workflow rules along with local systems concurrent transaction processing requirements (i.e., local serializability, rigorousness, etc.), so it is reasonable to ask whether a successful marriage of WFMS with TP monitors and more generally with DBMS's is possible. Finally, recovery aspects commonly attributed to a transaction-processing environment need new underpinnings in the context of workflow management, due, for example, to the fact that compensation is often more appropriate than simple undoing of actions.

The seminar clearly emphasized the view that workflow management is a core issue of computer science and emerges from the transaction area; this justifies the term "transactional workflows." It explored transaction-related concepts for workflow management (adequate modeling, correctness, scheduling, concurrency control, recovery), but also touched upon related questions. The following is an abstract of my introductory talk given on the first morning of the seminar:

The transaction concept has been around for many years, primarily in the database area. It has undergone a considerable evolution from primitive read/write operations to powerful operations described in a high-level script language. A recent idea, to be studied in this seminar, is to transfer the transaction concept from data-centric settings to process-centric ones, as brought along by workflow management. To get the seminar going, I first survey workflow issues and applications, then summarize the transaction concept, and finally try to give initial pointers on how the two fields could grow together, or what obstacles are present. I end with a list of questions, to which we hopefully find answers during this week.

References

[1] Y. Breitbart, A. Deacon, H.-J. Schek, A. Sheth, G. Weikum: Merging Application-Centric and Data-Centric Approaches to Support Transaction-Oriented Multi-system Workflows; ACM SIGMOD Record, September 1993.

[2] A.K. Elmagarmid (ed.): *Database Transaction Models for Advanced Applications*; Morgan Kaufmann 1992.

[3] D. Georgakopoulos, M. Hornick, A. Sheth: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure; Distributed and Parallel Databases 3, 1995, 119–153.

[4] D. Georgakopoulos, M. Hornick, F. Manola: Customizing Transaction Models and Mechanisms in a Programmable Environment Supporting Reliable Workflow Automation; GTE Tech. Report TC-0263-03-95-165, 1995.

[5] M. Hsu (ed.): *Special Issue on Workflow and Extended Transaction Systems.* IEEE Data Engineering Bulletin 16 (2) 1993.

[6] M. Hsu (ed.): *Special Issue on Workflow Systems.* IEEE Data Engineering Bulletin 18 (1) 1995.

[7] F. Leymann: Supporting Business Transactions via Partial Backward Recovery in Workflow Management Systems; Proc. BTW '95 Databases in Office, Engineering and Science, (Dresden, Germany, March 22-24, 1995), Springer 1995.

[8] F. Leymann: Transaktionskonzepte für Workflow-Managementsysteme; In: [12], 335–352.

[9] R. Obermarck (ed.): *Special Issue on TP Monitors and Distributed Transaction Management.* IEEE Data Engineering Bulletin 17 (1) 1994.

[10] M. Rusinkiewicz, A. Sheth: Specification and Execution of Transactional Workflows. In: W. Kim (ed.): *Modern Database Systems — The Object Model, Interoperability, and Beyond,* Addison-Wesley, 1995, 592–620.

[11] A. Sheth (ed.): *Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions*; Proc. NSF Workshop, Athens, GA, May 1996.

[12] G. Vossen, J. Becker (Hrsg.): *Geschäftsprozeßmodellierung und Workflow-Management — Modelle, Methoden, Werkzeuge*; International Thomson Publishing, Bonn 1996.

1.2 Transactional Workflows — A Contradiction?

by Hans-Jörg Schek

While for many it is no question that workflows should be transactional, it seems that some experts doubt that transactional workflows make any sense. There are even papers with claims that transactions and serializability are inappropriate or even undesired for workflows. For those, transactional workflows clearly is a contradiction. But why are workflows not transactions? Why not? What are transactions after all? What is serializability? Some define serializability as the result of 2PL on data pages. Is this the ultimate truth? What is correctness beyond serializability? Do we find answers by moving to semantic serializability? And to semantic atomicity?

We could make a little test and look at the following sentences in order to find out what the difference between transactions and workflows is. Here is the test. Please check the following definitions and answer with yes, maybe, or no:

A transaction is

- a sequence of reads & writes on pages or records with ACID property,
- a sequence of SQL statements with commit or abort at the end,
- a program with embedded SQL and control statements,

- a program with several (asynchronous) RPC and trigger executions,
- a script with several service calls to resource managers coordinated by a tp monitor,
- control and data flow between activities coordinated by a workflow engine.

The test is easy. One has passed it if the number of yes answers is two or more. Note that it is not the point to say that *every* sequence of SQL statements is a transaction or that *every* script with service calls must be a transaction. The important point is that it *can be* a transaction. It makes a lot of sense for the application developer to tell the system that a sequence of service calls *must be atomic* and/or that it *must be isolated*.

Classical ACID transactions bundle persistence, atomicity and isolation together. Clearly there is no inherent need to stay with this. It is not a shift of paradigm if we require that future transactions must offer these properties more independently. It must be possible to require atomicity without isolation for example, or isolation without atomicity, if this is possible at all. Clearly we need to investigate whether dependencies between these properties exist and what the restrictions are. We must stay on solid and well-understood grounds. What is correct compensation in case of early commit? To whom do I want to commit an activity and to whom should commits be deferred and how long? Can I do forward recovery after partial rollback without generating cascades, i.e. without undesired dependencies to other concurrent activities? What about alternative executions and dynamic changes of the execution? What does correct execution and controlled termination mean?

While these questions still require a lot of research, the point to make here is that many of these questions can be answered by looking into theoretical foundations from transaction research during the past ten years, such as work in [BBG89], [VHBS96], [WS92]. Although politicians require paradigm shifts we should not be driven be this and by fashion. We must further develop our understood theory and apply and generalize it. Our field needs soundness and continuity and less fashion waves.

References

[BBG89] Beeri, C., P.A. Bernstein, N. Goodman: A Model for Concurrency in Nested Transaction Systems, Journal of the ACM 36, No. 2, pp. 230-269, 1989.

[VHBS96] Vingralek, R., Hasse, H., Breitbart, Y., Schek, H.-J.: Unifying Concurrency Control And Recovery of Transactions With Semantically Rich Operations. In: Journal of Theoretical Computer Science, to appear 1996.

[WS92] Gerhard Weikum and Hans-J. Schek: Concepts and Applications of Multilevel Transactions and Open Nested Transactions. In: Database Transaction Models, Editor: Ahmed K. Elmagarmid, Morgan Kaufmann, 1992.

1.3 Organization of the Report

The meeting brought together 35 scientists from 5 different countries. During the week, 25 presentations were given with plenty of discussion time during and after each; in addition, ample time were reserved for discussions of special topics in small working groups. The remainder of this report contains the final program (Section 2), abstracts of the presentations (Section 3), working group summaries (Section 4), the complete list of participants (Section 5), and, following recent practice, a collection of each participant's URL (Section 6).

We felt that all participants enjoyed the workshop and the pleasant and constructive atmosphere of the Dagstuhl Castle, which this time was coupled with splendid weather throughout the whole week. We particularly wish to thank the Dagstuhl staff for ensuring that everything ran so smoothly.

2 Final Program

The program consisted of talks and several working groups which reported their findings on the last day. In addition, there was the usual Wednesday afternoon excursion, and a presentation and demo of the DB & LP Web Server of Trier. In detail, the program was as follows:

- Monday, July 15, 1996, 9–12.15pm
  Chair: F. Leymann

  1. Frank Leymann:
     Opening Remarks, Participants' Introduction
  2. Gottfried Vossen:
     Transactional Workflows: An Introduction
  3. Hans-Jörg Schek:
     Transactional Workflows — a Contradiction?

     Break 10.45–11.15am
  4. Gustavo Alonso:
     Transaction Processing and Workflow Management

- Monday, July 15, 1996, 1.45–5.30pm
  Chair: G. Alonso

  1. Frank Leymann:
     Semantical Recovery and Crash Recovery in WFMS
  2. Dieter Roller:
     Workflow-based Applications

     Break 3.30–4pm
  3. Betty Salzberg:
     Simplified and Formalized DSDT: No Heterogeneity, No Loops, just Log Records
  4. C. Mohan:
     Exotica: A Workflow Management Project at IBM Almaden Research Center

- Tuesday, July 16, 1996, 9–12.30pm
  Chair: G. Weikum

  1. Johann Eder:
     Workflow Transactions in Panta Rhei
  2. Mathias Weske:
     Event-Based Modeling of Workflow Executions

     Break 10.30–11am

3. Klaus Schwab:
   A Case Study: Transactional Workflows in the Cooperative System PlanKo — Problems and Possible Solutions
4. Wolfgang Deiters:
   Transactions in the FUNSOFT Net Approach for Process Management

- Tuesday, July 16, 1996, 2–5.30pm
  Chair: C. Mohan

1. Christoph Bussler:
   Reliable and Scalable Execution of Workflows in the Distributed WFMS MOBILE
2. Gerhard Weikum:
   Considerations on Scalability and Availability in the MENTOR Architecture

   Break 3.30–4pm
3. Gerti Kappel:
   A Transaction Model for Handling Composite Events
4. Thomas Rose:
   Document Management: Technology Means for Workflow Implementation

- Tuesday, July 16, 1996, starting at 7.30pm
  Working Groups

- Wednesday, July 17, 1996, 9–12.30pm
  Chair: G. Kappel

1. Gerhard Chroust:
   (a) Workflow — A Survey in the German-Speaking Countries
   (b) Process Enactment in Software Development, Office Automation, and Flexible Manufacturing — A Comparison
2. Andreas Geppert:
   Design and Implementation of Process-Oriented Environments

   Break 10.30–11am
3. Working Groups cont'd

- Wednesday, July 17, 1996, 1.15–6pm:
  Trip to Trier: guided tour of the city

- Wednesday, July 17, 1996, 7.30–9pm:
  Demonstration of the DB&LP Web Server at the University of Trier
  (by Michael Ley, U Trier)

- Thursday, July 18, 1996, 9–12.30pm
  Chair: J. Klein

  1. Peter Muth:
     The Cooperative Activity Model
  2. Andreas Oberweis:
     Petri-Net-Based Workflow Management

     Break 10.30–11am
  3. Heinrich Jasper:
     Time Issues in Advanced Workflows
  4. Tony Bonner:
     Concurrent Transaction Logic

- Thursday, July 18, 1996, 2–5.30pm
  Chair: M. Weske

  1. Rainer Unland, Ulrich Wanka:
     Consistency Issues in Multi Agent Systems
  2. Bernd Mitschang, Norbert Ritter:
     Cooperative DesignFlows in CONCORD

     Break 3.30–4pm
  3. Working Groups cont'd

- Thursday, July 18, 1996, starting at 7.30pm
  Working Groups cont'd

- Friday, July 19, 1996, 9–12
  Chair: R. Unland

  1. Hans Schuster:
     Integration of (Transactional) Application Programs into Workflows
  2. Johannes Klein:
     Active Messaging — towards simple workflows

     Break 10.30–11
  3. Findings Reports by the Working Groups:
     What Have We Learned?

3 Abstracts of Presentations

The following abstracts of presentations appear in alphabetical order of speakers' last names. The titles showing up in this section may vary slightly from the titles of the corresponding presentations as reported in the previous section.

Research Issues in Workflow Management Systems

Gustavo **Alonso**, ETH Zuerich, Switzerland

Workflow Management Systems have enjoyed considerable success in the last few years. They are, however, victims of their own success. Designed for small groups and small loads, and without a clear understanding of the environment where they were going to be deployed, workflow products are far from being able to meet the requirements set by current applications. Part of the problem is the goals of workflow systems, which may may not be within technological reach for a number of years. There is then the question of whether current systems will survive long enough to become established tools or they will disappear with their main ideas being discarded or, with some luck, incorporated in a different set of products/tools.

From a research point of view this situation creates a number of uncertainties that make it very difficult to decide which areas are worthwhile research topics and which areas are simply fashionable topics but non-issues from a scientific and sometimes also from a practical point of view. As part of the workshop the goal will be to determine how much research needs to be done in this area, probably very little, and how much engineering, most likely a lot. Additionally, a number of issues need to be studied in terms of the role they play in workflow systems.

Concurrent Transaction Logic

Tony **Bonner**, University of Toronto, Canada

In previous work, we developed Transaction Logic (or TR), which deals with state changes in databases. TR provides a logical framework in which elementary database updates and queries can be combined into complex database transactions. TR accounts not only for the updates themselves, but also for important related problems, such as the order of update operations, non-determinism, and transaction failure and rollback. In this talk, we propose Concurrent Transaction Logic (or CTR), which extends Transaction Logic with connectives for modeling the concurrent execution of complex processes. Concurrent processes in CTR execute in an interleaved fashion and can communicate and synchronize themselves. Like classical logic, CTR has a "Horn" fragment that has both a procedural and a declarative semantics, in which users can program and execute database transactions. CTR is thus a deductive database language that integrates concurrency, communication, and updates. All this is accomplished in a completely logical framework, including a natural model theory and a proof theory. Moreover, the framework is flexible enough to accommodate many different semantics for updates and

deductive databases. Finally, the proof theory for CTR has an efficient SLD-style proof procedure. As in the sequential version of the logic, this proof procedure not only finds proofs, it also executes concurrent transactions, finds their execution schedules, and updates the database. A main result is that the proof theory is sound and complete for the model theory. Besides specifying database transactions, potential applications of CTR include view updates, active databases and workflow management. Papers on TR and CTR, a prototype implementation of TR, and a tutorial can be found at the following URL: `http://db.toronto.edu:8020/transaction-logic.html`

Scalable and Reliable Execution in the Distributed Workflow-Management-System MOBILE

Christoph **Bussler**, University of Erlangen, Germany

Workflow management is a relatively new area in the research community. The "gestalt" of workflow, i.e., its functional contents is not yet precisely defined. New application areas might require to add new functionality, on the modeling level as well as on the implementation level.

MOBILE as a comprehensive approach to workflow management respects this and is open in the following sense: the model of MOBILE, its language for defining workflow types, its software architecture as well as its implementation is built in such a modular way that extensions of functionality can be done easily, i.e., in a structured and straightforward way.

Concentrating on software architecture and implementation work achieved so far, this contribution emphasized the necessity of being able to adjust to requirements coming from the deployment environment. If a certain network topology and communication infrastructure is given, an implementation should be adjusted to that in such a way that it fits optimal. Since environments change, one single implementation is never able to cope with all equally good.

MOBILE therefore allows to be configured dynamically as needed. To support this the MOBILE system is modularized according to functional criteria. These modules stand by themselves and do not assume a certain environment. Because of this they can be configured in various ways (like in one or multiple) servers such that special circumstances can be dealt with easily in a networked environment.

Workflow 1994 — A User Survey

Gerhard **Chroust**, Kepler University, Linz, Austria

Workflow is a much-discussed topic in the press, but what is the real influence in industry? In order to find out, our department initiated in fall 1994 a survey in Austria, Germany and Switzerland based upon a questionnaire containing some 70 questions including subquestions. Some 370 questionnaires were returned (5The answers — even if

strictly speaking statistically not representative — showed some interesting aspects of current use of workflow tools. The following observations could be made:

- Companies, on the average, have a good penetration with PCs and they are usually interconnected.
- Despite the fact that 50% of the documents are created on the computer, most of them are still archived and transported in paper form. Transport is largely done by persons (even if a computer network is in place), in smaller companies by the person having worked on the document, in larger companies by managers.
- Numerous copies are made, their number increases with company size.
- Workflow products seem to be used by small coherent groups only, communication with the rest of the company is by conventional means.
- The concept of business process is not universally understood.
- Workflow products, when used, are considered very good by users.
- The return on investment is seen as satisfying.
- The use of workflow products has some useful influence on customer relations.
- Imaging and archiving are seen as the main functionality.

Summing up it seems that workflow products are — in the industrial world — still in the experimental phase. Given the general satisfaction, they should make their way into general usage.

Process Models in Different Areas: Comparing Paradigms

Gerhard **Chroust**, Kepler University, Linz, Austria

In many areas we see socio-technical processes, i.e., processes where humans and machines cooperate in different roles. The processes in general need computer support. Currently three areas receive considerable attention:

1. Software development: supporting work by a process model and in enacting environment
2. Workflow: supporting office work and the flow of documents by a supporting environment.
3. Work cell design (CIM): defining and enacting process descriptions which define the sections and behavior of the robots, machines and transport facilities within the cell.

Comparison is done on

- the character of the process,
- the user's role perception,
- the documentation needs of the process,
- the process' security requirements,
- the multiplicity of the processes.

We argue that despite some differences the same enactment paradigm can be used.

Modeling Transactions in the FUNSOFT Net Approach to Process Management

Wolfgang **Deiters**, FhG ISST Dortmund, Germany

In our approach we aim at supporting various phases of what we call the "life-cycle of business processes." The phases are

- a modeling phase, the definition of explicit descriptions of the processes,
- an analysis phase, the validation and verification of models in order to detect problems, errors and bottlenecks and to optimize the model, and
- an enactment phase, in order to support the processes to be carried out (workflows) by appropriate IT-tools.

In order to provide support for these phases FUNSOFT nets, a high level Petri net language, are being used in an approach that distinguishes between different representations. On a user level representation different views emphasizing different aspects of process models are offered. These view representations are mapped and integrated onto an internal representation for analysis and enactment purposes.

An application of FUNSOFT nets revealed further requirements in order to appropriately describe and manage process models. These requirements are: - the identification of an atomic sphere for certain activity schedules within a process model (described by net parts in a FUNSOFT net) - the possibility to deal with conditions that make it necessary to stop and roll back process parts (that means to abort and roll back the firing of net parts For supporting these requirements the FUNSOFT nets are being extended by the concept of FUNSOFT net transactions.

In order to keep the resulting nets comprehensible in size (in terms of net nodes) new net elements (non-refined and refined transaction agencies) have been introduced on the user level representation. These transaction agencies identify the atomic sphere (the net part that should be run under transactional properties). Second, startup and termination conditions are introduced in order to distinguish objects that are needed for

starting or terminating a transaction from those objects that are seen as intermediate inputs or outputs. For the latter objects the isolation property is released. Furthermore, they encompass definitions for abort and recovery. A transaction can be aborted due to time constraints (if a specified time limit is exceeded), due to internal abortion trigger (conditions defined in the process model), or due to external abortion trigger (abort by an authorized user of the process management system).

The required behavior of FUNSOFT transactions is achieved by mapping transaction agencies onto nets in the internal process model representation. That means for each transaction agency new net parts are generated that ensure the required transactional properties. These net parts are integrated with those net parts from the user level representation defining the remaining process schedule. By that the resulting nets are kept comprehensible for understanding process models on the user level representation while the required transactional properties are ensured on the internal process representation. A first version of the transaction concept for FUNSOFT nets has been implemented in the CORMAN prototype for process management.

Workflow Transactions in Panta Rhei

Johann **Eder**, University of Klagenfurt, Austria

Transactional features for Workflow Management systems are needed for easier modeling and executing business processes. For the consideration which transactional features should be provided by a workflow management system we have to consider the division of work between the workflow management system itself, the application programs invoked by the workflow management system and the user. In particular, we found recovery an important issue. We extend the definition of workflows with some information about the compensatability of activities and generate from the workflow for usual situation a set of alternate processes which are executed in the case of exceptions. Further transactional features we introduce are the management of shared resources, the classification of data with respect to their changeability and the management of time. Several of these features have been implemented in our prototype system Panta Rhei.

Design and Implementation of Process-Oriented Environments

Andreas **Geppert**, University of Zurich, Switzerland

Cooperative process-oriented environments are complex, heterogeneous, distributed systems whose behavior is defined by process models. Examples for such environments are workflow systems and process-centered software development environments.

We propose an executable software architecture model to structure and describe this kind of environment. The major constructs in this model are brokers and services. Services model some functionality of the system or its components, they can for instance refer to activities to be executed in a workflow. Brokers are reactive components being able to react to simple events (such as requests for a service, a reply to a service request,

etc.), or composite events (e.g., the termination of multiple concurrent activities). Brokers can, e.g., represent processing entities in a workflow or wrap an external system. Thus, they do not only reflect the static architecture, due to their reactiveness they are also able to execute workflows.

A software architecture (in terms of brokers and services) is made executable through a low-level event engine, called EvE. EvE supports event-condition-action rules (ECA rules). These rules are used to implement the reactive behavior of brokers. The distributed multi-server architecture of EvE makes it possible to execute workflows in geographically distributed environments.

Time Issues in Advanced Workflows

Heinrich **Jasper**, University of Oldenburg, Germany

Workflow management systems schedule tasks in accordance with previously known process structures. Additionally, advanced workflow applications have to deal with parametric processes and exceptional situations. Using active database technology to implement such applications results in mature demands for their respective time handling capabilities. This regards both, the possibilities of time specification as well as a sophisticated time management within the rule execution mechanism. It turns out, that neither the management of time in conventional database systems nor the handling of time in most active database systems are sufficient for controlling workflows. This is due to the following problems that have to be solved:

1. Lack of a semantically well founded notion of time for both, data management as well as rule processing in active databases.

2. Lack of a uniform sub-language for the specification of time constraints for both, data and rules.

3. In order to support reliable rule execution for workflow management, a relaxed transaction model must be used. Here, the recovery of the active component of the DBMS, especially of time-events is a crucial problem which has to be tackled.

As an overall result, it turned out that techniques fulfilling soft real-time requirements are sufficient in the aforementioned areas.

A Transaction Model For Handling Composite Events

Gerti **Kappel**, University of Linz, Austria

Rule-based (re)active systems are a commonly accepted solution in the area of non-standard applications in order to express an event-driven and constraint-driven system environment. Several attempts have been made to integrate active concepts into object-oriented databases and to extend active knowledge models to gain more and more expressive power and flexibility. Unfortunately, execution models of active systems do not

fully exploit all the advantages of the provided knowledge models. Among the most challenging research problems is the development of a transaction model in the presence of so called composite events. Our approach of multi-parent sub-transactions tries to fill this gap. Multi-parent sub-transactions extend the well-known nested transaction model by allowing multiple transactions to start a sub-transaction in cooperation. In particular, we will discuss the impacts of our extensions on the locking protocol of the original nested transaction model.

Active Messaging

Johannes **Klein**, Tandem Computers Inc., USA

Data management, request processing and, messaging are the cornerstones of commercial data processing. During the 70's and 80's the research community developed modern concepts for data management and request processing. Examples are SQL and RPC style invocation mechanisms. Messaging is the focus of the 90's. Research on workflow management and mobile agents are the result. As with most efforts initial concepts are complex and difficult to use. Especially in workflow management complex programming paradigms are dominant. Acceptance of workflow management systems will depend on a simple but powerful programming model.

Active messages introduce an approach based on task containers. Task containers represent individual steps as well as complete business processes. Controlflow is part of the definition of a task container. Business processes are defined as a composition of task containers. Predefined task containers are provided as templates for repetitive tasks. Task containers can be modified at any time during execution of a business process to cope with exceptions. Once a task container has been instantiated it moves from service to service. All services are executed with transactional protection.

Task containers define a simple but powerful programming model for workflow applications. Domain specific constraints will be implemented by specialized task containers. Java and similar technologies will allow the exchange of task containers in heterogeneous networks and across organizational boundaries.

Crash Recovery and Semantical Recovery in WFMS

Frank **Leymann**, IBM Germany

Production workflow systems deal with the enactment of business process models of high business value for enterprises. Traditionally, corresponding application systems are implemented via transactions. So, it does not surprise that users of production workflows ask for "transactional features" supported by WFMS. First, the engine itself must be translated in the sense of providing Phoenix behavior (persistent states, forward recovery, etc.). Secondly, since WFMS stimulate reuse ("flow independent" activity implementation) of programs, these programs must not assume transaction boundaries; these boundaries are specified and established at the workflow level ("atomic spheres").

In addition, since workflows represent (very) long lasting computations the encompassed steps of which are often not reversible, a partial compensation-based rollback mechanism has to be provided ("compensation spheres"). This positions WFMS as the next generation of transaction management systems.

Exotica: A Project on Workflow and Advanced Transaction Management

C. **Mohan**, IBM Almaden Research Center, San Jose, USA

The Exotica project has focused on exploring advanced transaction management concepts and many issues relating to workflow management. Our work has been done in the context of IBM products CICS, MQSeries and FlowMark, and Lotus Notes. Many papers have been written (see http://www.almaden.ibm.com/cs/exotica) which relate to the following topics: scalability, availability, exploitation of transactional messaging for distributed workflow management, mapping advanced transaction models like Sagas and Flex transactions to FlowMark, providing high-level process definition capabilities on top of Lotus Notes, supporting disconnected mobile users of FlowMark, etc. A prototype has been built which modified FlowMark to allow ready work items to be checked out and then be executed at a time when the client is not necessarily connected to the server. In the future, we plan to provide more advanced functionality in this area (e.g., checking out a ready activity and some follow-on non-ready activities). Currently we are also exploring better integration of FlowMark with desktop applications (e.g., OLE enablement) and the Internet. Early on in the project we also worked on compensation concepts for legacy applications which involve TP monitors like CICS. Recently, we have also been analyzing the use of DB2 by FlowMark instead of ObjectStore as the repository for process control data.

The Cooperative Activity Model

Peter **Muth**, University of the Saarland, Germany

With the emergence of cooperative applications it turned out that traditional transaction concepts are not suitable for theses scenarios. We propose a cooperative activity model that provides the required transactional properties for cooperative scenarios. The talk focuses on the problem of merging actions of different co-workers, executed in isolation, into a single, legal history. We investigate the usability of well-known conflict predicates like read/write conflicts and commutativity as the basis for dependency relations which define a correct merge. We finally propose a new dependency relation which is less restrictive than the other relations investigated, and is defined independent of actual histories.

Petri Net Based Workflow Management

Andreas **Oberweis**, University of Frankfurt, Germany

The goal of workflow management is to support business processes by using information and communication technology such as workflow and database management systems. A business process model usually describes a business process from an application oriented perspective and includes a description of aspects such as ordering of activities, temporal aspects, cost and quality issues, organizational constraints. Different semi-formal notations have been proposed for business process modeling. A workflow program on the other side is an executable description of what a workflow management system should do with respect to a certain workflow. It includes synchronization aspects, database accesses and interfaces to application software, hardware devices etc. Between business process model and workflow program there exists a "semantic gap" which can be bridged by formal, platform and implementation independent languages, e.g., Petri nets. It should be noted that in general there does not exist a 1:1 mapping between business process model, formal "conceptual" model and workflow program: some information is only contained in the business process model, some information only in the workflow program. Usually it is not possible to map one model completely automatically into another model. High-level Petri nets are closely related to databases: places can be interpreted as data types, a place marking as a database state, a transition as a type of database transaction and a transition occurrence as an instance of the respective type. Petri nets are extensible in a way that arbitrary data models may be used to model places.

A Petri net specifies a type of workflow, a partial ordered execution of transition occurrences in the net describes an instance of a workflow type. The occurrence rule precisely describes, what transitions are enabled (i.e. may occur) in a given state and how a transition occurrence changes the state. Additionally, spheres of atomicity or isolation can be defined in a given Petri net, and mechanisms to guarantee these spheres can be included as refinements of the given net. Petri net interpreters check for a given initial marking, what transitions are enabled and "fire" selected enabled transitions. They can be used for net simulation to validate a given workflow schema or to do "What-if-analyses" at workflow runtime. A net interpreter can also be used as a simple workflow engine. However, this requires interfaces to application programs, database system(s), hardware devices and workflow users. Petri net based workflow models can also be used as a formal basis for workflow distribution in a networked environment: they allow fragmentation of workflows and allocation of workflow fragments in a natural way. Data and workflow fragmentation and allocation can be handled in an integrated formalism.

A Petri net interpreter can also be used to control workflows in (standard) application software systems (and by this contribute to the systems' flexibility). Finally a Petri net interpreter may also be used as a dynamic extension of a relational database management system on top of transaction manager, query optimizer etc., leading to some kind of "database with integrated workflow support."

Cooperative Design Flows in CONCORD

Norbert **Ritter**, University of Kaiserslautern, Germany [1]

Amongst the most important components in concurrent design/engineering environments is a flexible and sophisticated activity management. Designflow management and cooperation control are crucial issues of activity management. An adequate support requires means for guiding designers through design tasks as well as mechanisms for coordinating the work of several designers involved in a common design process. The latter, in turn, demands flexible coordination and cooperation concepts supporting both, a pre-planned as well as a free cooperation among designers.

The CONCORD (CONtrolling COopeRation in Design environments) provides activity-control concepts for design environments by integrating transaction protection of design-data manipulations, designflow management, as well as cooperation control. This presentation focuses on the interplay of cooperation control and designflow management and discusses how the above mentioned flexibility requirements are met in CONCORD.

Workflow-Based Applications

Dieter **Roller**, IBM Germany

The structure of applications is changing from monolithical applications managing their own data and control flow to workflow-based applications where the control- and data flow is described to and managed by a workflow management system. This type of application is by nature distributed and executing in a heterogeneous environment. A new methodology called Process-based Case supports the creation, debugging, testing, and monitoring of workflow-based applications. The underlying two-level programming, where process modeling is supported by the WFMS build-time and activity implementation is performed by Visual Builders, allows the rapid construction of applications. In a final step, the activity implementations are represented by the invocation of methods against business objects.

Document Management: Driving Technology for the Implementation of Business Processes

Thomas **Rose**, FAW Ulm, Germany

This position statement addresses workflow management from an application-oriented perspective. Workflow management is intended to improve an organization's performance in terms of quality of services, customer responsiveness, and traceability of operations, be it for eco-auditing or other certification purposes. Yet, companies that have not reached a sufficiently mature level of organizational management are not familiar

[1] Joint Work with Bernhard Mitschang, Technical University of Munich, and Theo Härder, University of Kaiserslautern.

with the concept of business processes. They cannot name and describe their processes. Hence, they are not well enough prepared to start electronic workflow management. They, however, face huge amounts of documents, which ought to be consistent and float around the enterprise. In these cases, document management has proven to be the key element to disembark on designing and eventually implementing business processes: introduce electronic document handling facilities first, and then implement workflow applications on top of it.

Simplified and Formalized DSDT: No Heterogeneity, No Loops, Just Log Records

Betty **Salzberg**, Northeastern University, Boston, USA

DSDT (Durable Scripts containing Database Transactions) (ICDE 1996, pp. 624-633) is a proposal for making long-running activities durable. These activities are described with a script containing code units (CUs). Code units are either deterministic (DCUs) or logged (LCUs). Deterministic code units (DCUs) can be replayed during recovery from a failure. Code units which cannot be replayed (LCUs) include calls to transactions, user input, or received messages from remote locations, for example. A standard DBMS is used for logging results of LCUs. The most recent stable script state (SSS) is placed in the standard log checkpoint record. The state of the script can be recovered by obtaining the SSS from the log checkpoint and replaying DCUs and obtaining information from log records pertaining to the script which are in the log after the checkpoint. This can be done in the same forward pass made by a standard recovery manager after system failure.

Integration of (Transactional) Application Programs into Workflows

Hans **Schuster**, University of Erlangen-Nuremberg, Germany

Application programs are a substantial asset of an organization. The introduction of workflow management technology must provide means and mechanisms to integrate existing application programs which are not aware of workflow management systems in the same seamless manner as newly written, workflow-aware applications. This work presents a general approach for integrating workflow-unaware application programs into workflow management systems. Access to application programs within workflows is done using so-called workflow applications. Workflow applications are internally represented by workflow application objects. These provide a unique interface for the workflow management system to invoke external programs and build an abstraction from implementation details and characteristics of application programs (e.g., calling mechanism). Thus, the complexity of the workflow management system is reduced. Our concepts enable the integration of application programs which are implemented on top of heterogeneous operating systems or base services into our prototype workflow management

system Mobile. Furthermore, we support the execution of several transactional application programs within a single transaction controlled by the workflow management system.

Usage of Transactional Concepts in the Cooperation Management System PlanKo: Requirements, Problems and possible Solutions

Klaus **Schwab**, University of Bamberg, Germany

The goal of the PlanKo Cooperation Management System is to integrate different CSCW applications and workflows within a business process. Integration becomes necessary because most of the CSCW systems and workflow systems act as specialists. They only support a specific type of cooperation or communication instead of taking care of various cooperative needs of real-life work-groups involved in a business process. Three different levels of integration can be identified, which result in the three layer architecture of the PlanKo Cooperation Management System.

The Basic Data Storage Layer provides access to data jointly used by different cooperations of different types. Different CSCW systems and a workflow management system have been implemented at the Cooperative Application Layer. The Cooperation Access Layer enables cooperating people to be aware of their involvement in different cooperations and allows to manipulate it. On each layer integration components have been added: The Cooperative Context Control Component manages the access to data objects by cooperative applications. The Action Flow Control Component synchronizes waiting dependencies between different activities (steps) within different cooperations. The Cooperation Manager provides a common interface for the creation and the management of cooperations. Introduction of high level transaction concepts arise several problems within this environment. Isolation for examples is not an appropriate concept, for the results of work done in a single application should be externalized as soon as possible. An automatic rollback may therefore result in inconsistent data. Furthermore it is not clear, how "to roll back" more than one single workflow for there exists various dependencies between workflows. Therefore the only solution in respect to data sharing considered up to now is a specialized version management system.

Consistency Issues in Multiagent Systems

Ulrich **Wanka**, University of Muenster, Germany[2]

The aim of the talk was to identify requirements for a Customized Transaction Management (CTM) for Multiagent Systems (MAS). MAS is a subfield of Distributed Artificial Intelligence (DAI) that investigates how a set of different autonomous problem solvers can be coordinated in a way that they are able to cooperate in an intelligent way. The workflows or problem solving processes within such a system require different forms of

[2]Joint work with Rainer Unland, University of Essen, Germany.

cooperation. We differentiate between quality-sharing, task-sharing, and result-sharing. Whereas the first two forms can be implemented by nested transactions, the latter requires a fully cooperative environment (i.e., no isolation, early visibility of results). Since robustness is of uttermost importance for such kind of systems we suggest a partial rollback strategy supplemented by compensation. The question of how autonomy of the agents interferes with these methods remains to be investigated.

Considerations on Scalability and Availability in the MENTOR Architecture

Gerhard **Weikum**, University of the Saarland, Germany

The MENTOR project aims to reconcile a rigorous workflow specification method with a distributed middleware architecture as a step towards enterprise-wide workflow management. The project uses the formalism of state and activity charts for specification. For scalability, a workflow is partitioned into distributable components that can be assigned to different workflow servers, depending on the performance requirements and organizational decentralization of the application. Fault tolerance issues are addressed by using a TP monitor for communication between distributed workflow servers and by maintaining an additional log of workflow state information at each server. High availability can be ensured by optionally replicating workflow servers at different sites, with the degree of replication chosen according to the required availability level.

Event-Based Modeling of Workflow Executions

Mathias **Weske**, University of Münster, Germany

Workflow executions have so far mostly been treated in technical terms like client-server communication or the use of persistent message queues. However, a formal and system-independent way to model workflow executions is suited to describe and reason about workflow executions. In this talk we propose an event-based model that regards a workflow execution as a distributed computation, focusing on the events and their ordering that occur during the workflow execution. Hence, workflow executions are represented by event structures that can be analyzed w.r.t. correctness properties, like the compliance of a workflow execution with its specification. Finally, we discuss the use of our approach in the dynamic modification issue, i.e., to describe and reason about workflow executions in presence of dynamically changing workflow models.

4 Working Groups

During the week four different working groups were formed, each of which met several times to discuss particular issues in depth. Following are the summaries of these groups, which were presented on the last morning of the seminar.

4.1 Concurrent Executions

The basic subject of this working group was the question of what are reasonable correctness criteria for concurrent executions of multiple workflows. The major questions raised and hypotheses issued for this working group during an opening plenary session were the following:

- The process model is the specification of correctness, at least for single processes or workflow instances, not for parallel processes. Therefore, correctness for single workflow instances is the responsibility of the application programmer, and as a consequence it may be the case that *all* interleaved executions of workflows are correct.
- What are good examples for situations or scenarios in which correctness (of concurrent executions) really is an issue for workflows?
- What is essentially meant by correctness?
- Is "workspace serializability" the right (or at least a good) notion?
- Since there is more than a single shared resource in a workflow execution context, it is reasonable to distinguish static from dynamic constraints. Are such (concurrency) constraints needed for compensation issues?
- Are "spheres of isolation" the right concept? Are they independent of spheres of atomicity?
- Is serializability more restrictive that other correctness criteria, or is it more liberal? Is serializability actually what we want, or can be get away with something simpler (e.g., monitors, semaphores)?
- Can there be correctness by controlling the data, or is it the actions which should be controlled?

As could be expected, not all of these questions could be discussed or even answered during the meetings of the group. The following is a summary of what *did* come out and which (in part preliminary) conclusions were reached:

1. A reasonable view of the situation present in the concurrent execution of workflows distinguishes two levels: The upper level is represented by the workflow specification and the application design, while the lower level comprises the database(s) and its/their transactions. This assumes that workflow executions access databases by

means of transactions; since transactions operating on databases need synchronization, they get this through the respective DBMS mechanisms. It follows that, at the lower level, concepts like serializability remain in effect.

On the other hand, workflows (as entities of a higher level of abstraction) have to maintain business rules and dependencies driven by business goals, referring to accesses applied to shared enterprise data. If business rules are part of a given specification, which is often the case, no extra mechanism is required to enforce them. If they are not, a reasonable approach is to distinguish *global* consistency (at the upper level) from *local* consistency (at the lower level); a desirable correctness criterion would provide conditions under which local correctness (such as serializability) implies global correctness (w.r.t. business rules). Obviously, business rules create dependencies between workflows, and it may even be reasonable to distinguish business *rules* from business *requirements*.

The situation just described can be further refined according to whether it suffices to keep the overall situation consistent (e.g., *a bank giving credit should not go bankrupt*), or whether each individual workflow has to maintain an enterprise rule (e.g., *each bank workflow earns money for the bank*). An open question is whether there are abstractions which allow valid statements for a sufficiently large class of applications.

2. Since workflow transactions are typically long-running activities, atomicity and locking are no longer applicable. In this situation, "atomic spheres" may sometimes help, but generally compensation will be required to recover from a faulty situation. To enforce local, data-oriented dependencies (as found in databases), synchronization could hence alternatively be left to the objects themselves: objects would have to be aware of accesses to them and the correct sequencing for these; the appropriate paradigm could be that of "agents," and a relevant formalism to describe and investigate this approach could be automata, possibly augmented by synchronization means such as timestamps.

3. With respect to recovery, it occurred to the group that lower-level transaction synchronization also requires failure handling, for being able to properly react to transaction aborts or systems errors, whereas higher-level workflows and applications may even tolerate failures. The latter particularly applies when a workflow specification comprises detailed exception handling and treatment of failures and abnormal situations.

4. The examples discussed by the group for getting a better understanding of correctness issues included banks and private homes (credits and sales). These examples indicate that there exist synchronization problems in applications for which *explicit* solutions have been developed. For example, if someone tries to get credit based on his house and at the same time tries to sell the house to another person, the local central registry for house ownership will detect the incompatibility between these two actions and prevent the execution of one of them. Here, the

explicit solution is that registry, which has been invented a long time ago for exactly the purpose of synchronizing such conflicting actions. In workflow terms, contacting the registry is an action included in credit approvals or house transfers after a sale. In other words, these "synchronization units" are an integral part of the underlying application and hence contained in a corresponding workflow modeling.

5. The above does **not** imply that there are no synchronization problems left in concurrent workflow executions. Indeed, the considerations so far implicitly referred to workflows which can be completely specified in advance. However, there is an increasing number of situations in workflows are *ad-hoc*, i.e., not specified at all before their execution is launched. For such situations, the above may not be applicable.

(Summary by G. Vossen; participants were A. Brayner, J. Ebert, J. Eder, P. Muth, N. Ritter, G. Vossen; comments or questions were also contributed by G. Alonso, T. Bonner, G. Chroust, J. Klein, F. Leymann, B. Salzberg, H. Schek, G. Weikum.)

4.2 Recovery and Error Handling

We assume the following architecture: A workflow *engine*, the top layer, makes calls to several *tasks* at the lowest layer, and a durable queue maintains information about these calls at a middle layer. A database stores information about the state of the workflow.

The following failures can occur:

1. The engine can fail.

2. A task can return: fail, abort, success, or can return nothing, where "fail" means the task was canceled by its user, "abort" includes task/system failures such as deadlocks, beyond the control of the user.

There are various kinds of tasks, including

- manual/interactive tasks with the "user" involved, e.g., filling out forms, editing,
- purely manual, e.g., a robot in an assembly line,
- automatic (machine only), e.g., database transactions.

Possible reactions to the various kinds of failures, depending on the type of task involved are the following:

1. If the workflow engine fails, the state is recovered from the workflow database. If some tasks were in progress, the engine can wait for them to respond or can inquire about their status.

2. If a task fails it is up to the task to do any partial recovery such as resuming an editing session at the most recent saved version. The workflow engine cannot help for this, but it can be programmed to do various actions when a task fails. This may trigger some manual intervention, automatic intervention or the failure can be ignored. Alternatively, the task may be retried N times or there may be a timeout before other actions are taken.

3. Automated tasks such as database transactions may be undone if there is a failure, and then tried again. It may not be possible to undo manual actions. It may not be desirable to undo user-involved (manual interactive) tasks such as filling out a form; for such tasks, the goal is to disturb the user as little as possible.

This working group ended by discussing several types of errors and ways those errors could be handled.
(Summary by Betty Salzberg; participants were Walter Liebhart, Ute Masermann, Gerhard Chroust, Claus Hagen, Betty Salzberg, Klaus Schwab, Rainer Unland, Ulrich Wanka.)

4.3 Event-Driven Workflow Execution

In this working group, we have discussed the use (and usefulness) of "events" for workflow modeling and execution. In the very basic sense, an *event* (occurrence) is a happening of interest either in the system itself or in its environment. In many areas (including active databases, extended transaction models, distributed systems, software architecture, user interfaces), the general notion of event has been used to, e.g., describe the behavior of systems and prove correctness properties of distributed systems and algorithms. Thus, we feel that it is worthwhile to consider events for several aspects of workflow management.

We considered two areas in more detail:

- specification and semantics of workflows,
- execution and evolution of workflows.

With respect to the first topic, we agreed that "events" (in the sense of *event-condition-action rules* (ECA rules) as found in active databases) are not a useful concept for specifying workflows. This is due to the fact that the resulting set of ECA rules would be very large and, thus, hard to understand.

We then discussed events as a means to describe the semantics of workflows. Conceptually, we considered workflows on three different levels of abstraction. On the topmost level, workflows are specified in some formalism (Petri Nets, state-charts, dedicated specification languages). Such specifications might be (internally) represented as extended regular expressions over events, where the extension refers to constructs for parallelism.

On the bottom level we consider *event histories*. An event history is a partially ordered set of events and represents the order of events which occurred during a workflow. The correctness of a complete event history is defined by constraints which specify

ordering of events and coexistence of events. Ordering of events is either expressed as *delay*, i.e., $a > b$, (a is delayed until b occurred) or *deadline*, i.e. $b < a$ (b cannot occur after a occurred). The ordering constraints allow for the following event histories: $\{\}$, $\{a\}$, $\{b\}$, $\{b, a\}$. Coexistence of events is expressed by logical implication, i.e., $a \rightarrow b$. The following histories satisfy the implication: $\{\}$, $\{b\}$, $\{a, b\}$, $\{b, a\}$.

Event histories can be used to reason about the correctness of workflow executions. Consider the aforementioned regular expressions as a language definition defining the intention of a workflow (type). A concrete workflow execution is then considered as correct if the corresponding event history is a phrase of the language defined by the expression.

Correctness can be either proven after workflow execution (by checking a history against the expression), or it can be enforced during workflow execution (by ensuring that the event history conforms to the expression).

We further considered event-driven workflow execution, and compared it to other approaches (i.e., script-based approaches). The event-driven approach means that a workflow specification is compiled into a set of rules (e.g., ECA rules), while in many script-based approaches an interpretative approach is pursued. We concluded that event-driven workflow execution performed by some kind of active persistent system is supposed to be more efficient than an interpretative one for the several reasons.

- The workflow definitions are compiled into ECA rules, and due to the executability of these rules we expect a performance gain in comparison to interpretation of scripts.
- Detection of relevant events is performed by the engine and polling for event occurrences (as it is necessary in script-based approaches which operate on top of a database) is circumvented.
- Condition evaluation can be pushed into the engine and can be performed directly after event detection (thus process dispatches are avoided and active workflows can be scheduled more efficiently).

For practical cases, several solutions concerning the specification of ad-hoc workflows and evolution of workflow definitions still have to be addressed. In order to define ad-hoc workflows, it must be possible on the implementation level to define new ECA rules "on the fly". We shortly discussed self-modifying sets of ECA rules as means to handle workflow evolution as well as for flexible exception handling. We concluded that more work is necessary to understand the problems and possible solutions of workflow evolution in event-driven approaches (this certainly holds for script-based approaches, too), while we felt that the technical issues related to evolution can be handled in some way. (Summary by Andreas Geppert; participants were Heinrich Jasper, Gerti Kappel, Johannes Klein, Mathias Weske).

4.4 Scalability

The main difficulty when discussing scalability of workflow systems is that the components are not known, i.e., unlike in databases, in workflow systems it is still unclear

which are the key system components and their impact on performance and scalability. As a result, every product and prototype addresses the problem in an entirely ad-hoc manner in a process that resembles more debugging than optimization. Questions that need to be addressed by the research community are the overall architecture of a workflow system and the role of the database as a building block. For instance, it is possible that many of the scalability problems of workflow systems can be solved by using the appropriate database (replicated, scalable, high performance, maybe a parallel database and so on). The discussions in the work group were centered around this premise and, hence, several areas were identified in which research is needed:

1. Extend/export database functionality for workflow applications. Current databases do not provide the primitives needed in a workflow environment. It would not be very difficult to customize the database engine to the operations and characteristics of workflow tools so as to facilitate their integration. Using commercial, industrial strength databases, many of the scalability problems could be solved.

2. Workflow requirements are not standard. It is widely recognized that workflow requirements go beyond what is technically feasible nowadays and are different from traditional database applications. For instance, workflows tend to be heavily update oriented. It is therefore necessary to continue the research work in areas such as distributed systems, very large databases, and distributed information systems.

3. Scalability is determined by the database architecture. Judging from the experience of commercial products, many scalability problems arise from the integration of the workflow system with the database. From the modeling stages to the actual database access, workflow systems still lack a clear understanding of what are adequate design approaches and how they can be optimized. Scalability will remain a problem as long as there are no sound concepts in which to base the design.

4. Workflow model/workload as a scalability factor. Scalability is not always an issue. There are many workflow applications in which the nature of the environment allows to apply known techniques such as data partition and data replication to solve the scalability problem. It is an issue that deserves further analysis when these techniques can be used and how they can be adapted to workflow systems.

(Summary by G. Alonso; participants were Frank Leymann, C. Mohan, Dieter Roller, Christoph Bussler, Hans Schuster)

5 List of Participants

1. Gustavo Alonso
   Institut für Informationssysteme
   ETH Zürich
   ETH-Zentrum
   CH-8092 Zürich, Switzerland
   alonso@inf.ethz.ch

2. Anthony Bonner
   Dept. of Computer Science
   University of Toronto
   10 King's College Road
   Toronto, Ontario M5S 1A4, Canada
   bonner@db.toronto.edu

3. Angelo Brayner
   Institut für Wirtschaftsinformatik
   Universität Münster
   Grevenerstraße 91
   D-48159 Münster, Germany
   brayner@helios.uni-muenster.de

4. Christoph Bußler
   Universität Erlangen-Nuernberg
   Lehrstuhl für Datenbanksysteme
   (IMMD VI)
   Martensstraße 3
   D-91058 Erlangen, Germany
   bussler@informatik.uni-erlangen.de

5. Gerhard Chroust
   Universität Linz
   Systems Engineering and Automation
   Altenbergerstr. 69
   A-4040 Linz, Austria
   chroust@sea.uni-linz.ac.at

6. Wolfgang Deiters
   Fraunhofer-Gesellschaft
   Inst. f. Software & Systemtechnik
   Josef v. Fraunhoferstr. 20
   D-44227 Dortmund, Germany
   deiters@do.isst.fhg.de

7. Jürgen Ebert
   Fachbereich Informatik
   Universität Koblenz-Landau
   Rheinau 1
   D-56075 Koblenz, Germany
   ebert@informatik.uni-koblenz.de

8. Johann Eder
   Universität Klagenfurt
   Institut für Informatik
   Universitätsstrasse 65–67
   A-9022 Klagenfurt, Austria
   eder@ifi.uni-klu.ac.at

9. Andreas Geppert
   Universität Zürich
   Institut für Informatik
   Winterthurerstr. 190
   CH-8057 Zürich, Switzerland
   geppert@ifi.unizh.ch

10. Claus Hagen
    Institut für Informationssysteme
    ETH Zürich
    ETH-Zentrum
    CH-8092 Zürich, Switzerland
    hagen@inf.ethz.ch

11. Stefan Jablonski
    Universität Erlangen-Nuernberg
    Lehrstuhl für Datenbanksysteme
    (IMMD VI)
    Martensstraße 3
    D-91058 Erlangen, Germany
    jablonski@informatik.uni-erlangen.de

12. Heinrich Jasper
    Universität Oldenburg
    Fachbereich 10 Informatik
    Postfach 25 03
    D-26111 Oldenburg, Germany
    jasper@informatik.uni-oldenburg.de

13. Gerti Kappel
    Abt. für Informationssysteme
    Universität Linz
    Altenbergerstr. 69
    A-4040 Linz, Austria
    gerti@ifs.uni-linz.ac.at

14. Michael Kifer
SUNY at Stony Brook
Computer Science Dept.
Stony Brook, NY 11794-4400, USA
kifer@cs.sunysb.edu

15. Johannes Klein
Tandem Computers Inc.
10555 Ridgeview Court
Cupertino, CA 95014, USA
klein_johannes@tandem.com

16. Frank Leymann
IBM Entwicklung GmbH
Dept. 2366, IBM GSDL
Hanns-Klemm-Straße 45
D-71034 Boeblingen, Germany
frank_ley@vnet.ibm.com

17. Walter Liebhart
Universität Klagenfurt
Institut für Informatik
Universitätsstrasse 65–67
A-9022 Klagenfurt, Austria
walter@ifi.uni-klu.ac.at

18. Ute Masermann
MU Lübeck
Institut für Informationssysteme
Osterweide 8
D-23562 Lübeck, Germany
maserman@ifis.mu-luebeck.de

19. Bernhard Mitschang
TU München
Institut für Informatik
Orleansstr. 34
D-81667 München, Germany
mitsch@informatik.tu-muenchen.de

20. C. Mohan
IBM Almaden Research Center
K55/B1
650 Harry Road
San Jose, CA 95120-6099, USA
mohan@almaden.ibm.com

21. Peter Muth
Universiät des Saarlandes
Fachbereich 14 – Informatik
Postfach 15 11 50
D-66041 Saarbrücken, Germany
muth@cs.uni-sb.de

22. Andreas Oberweis
J.W. Goethe-Universitaet
Lehrstuhl f. Wirtschaftsinformatik II
Postfach 11 19 32
D-60054 Frankfurt, Germany
oberweis@wiwi.uni-frankfurt.de

23. Andreas Reuter
Universität Stuttgart
Institut für Parallele und
Verteilte Höchstleistungsrechner
Breitwiesenstr. 20–22
D-70565 Stuttgart, Germany
reuter@informatik.uni-stuttgart.de

24. Norbert Ritter
Universität Kaiserslautern
Fachbereich Informatik, AG DVS
Postfach 3049
D-67653 Kaiserslautern, Germany
ritter@informatik.uni-kl.de

25. Dieter Roller
IBM Entwicklung GmbH
Dept. 2366, IBM GSDL
Hanns-Klemm-Straße 45
D-71034 Boeblingen, Germany
droller@vnet.ibm.com

26. Thomas Rose
FAW Ulm
Helmholtzstr. 16
D-89081 Ulm, Germany
Rose@faw.uni-ulm.de

27. Betty Salzberg
Northeastern University
College of Computer Science
161 Cullianne Hall

Boston, MA 02115, USA
salzberg@ccs.neu.edu

28. Hans-Jörg Schek
Institut für Informationssysteme
ETH Zürich
ETH-Zentrum
CH-8092 Zürich, Switzerland
schek@inf.ethz.ch

29. Hans Schuster
Universität Erlangen-Nuernberg
Lehrstuhl für Datenbanksysteme
(IMMD VI)
Martensstraße 3
D-91058 Erlangen, Germany
schuster@informatik.uni-erlangen.de

30. Klaus Schwab
Universität Bamberg
Lehrstuhl für Wirtschaftsinformatik
Büro u. Verwaltungsautomation
Feldkirchenstrasse 21
D-96052 Bamberg, Germany
schwab@buva.sowi.uni-bamberg.de

31. Rainer Unland
Universität-GH Essen
Fachbereich Math. u. Informatik
Schützenbahn 70
D-45127 Essen, Germany
unlandr@informatik.uni-essen.de

32. Gottfried Vossen
Institut für Wirtschaftsinformatik
Universität Münster
Grevenerstraße 91
D-48159 Münster, Germany
vossen@helios.uni-muenster.de

33. Ulrich Wanka
Institut für Wirtschaftsinformatik
Universität Münster
Grevenerstraße 91
D-48159 Münster, Germany
dbulwa@helios.uni-muenster.de

34. Gerhard Weikum
Universiät des Saarlandes
Fachbereich 14 – Informatik
Postfach 15 11 50
D-66041 Saarbrücken, Germany
weikum@cs.uni-sb.de

35. Mathias Weske
Institut für Wirtschaftsinformatik
Universität Münster
Grevenerstraße 91
D-48159 Münster, Germany
weske@helios.uni-muenster.de

6 Participants' URLs

Alonso:
http://www.inf.ethz.ch/personal/alonso/alonso_page.html

Bonner:
http://db.toronto.edu:8020/people/bonner/bonner.html

Brayner:
http://wwwmath.uni-muenster.de/~dbis/Brayner

Bussler:
http://www6.informatik.uni-erlangen.de/~bussler

Chroust:
http://www.sea.uni-linz.ac.at

Deiters:
http://www.isst.fhg.de/pages/fotoDeiters.html

Ebert:
http://www.uni-koblenz.de/personen/EbertJuergen.entry

Eder:
http://www.ifi.uni-klu.ac.at
http://www.ifi.uni-klu.ac.at/~herb/workflow.html

Geppert:
http://www.ifi.unizh.ch/groups/dbtg/Staff/Geppert/andreas.html

Hagen:
http://www-dbs.inf.ethz.ch/chhomepage.html

Jablonski:
http://www6.informatik.uni-erlangen.de/dept/staff/jablonski.html

Jasper:
http://www-is.informatik.uni-oldenburg.de/ais.html

Kappel:
http://www.ifs.uni-linz.ac.at

Kifer:
http://www.cs.sunysb.edu/~kifer/

Klein:
http://www.tandem.com

Leymann:
http://www.software.ibm.com/ad/flowmark/exmn0mst.htm

Liebhart:
http://www.ifi.uni-klu.ac.at/home?walter

Masermann:
http://www.ifis.mu-luebeck.de/~ifis/staff/maserman.html

Mitschang:
http://www3.informatik.tu-muenchen.de/public//mitarbeiter/mitschang.html

Mohan:
http://www.almaden.ibm.com/cs/people/mohan
http://www.almaden.ibm.com/cs/exotica/

Muth:
http://www-dbs.cs.uni-sb.de/public_html/leute/peter/muth.html

Oberweis:
http://www.wiwi.uni-frankfurt.de/~oberweis/

Reuter:
http://www.informatik.uni-stuttgart.de/ipvr/as/as_home.html

Ritter:
http://www.uni-kl.de/AG-Haerder/
http://www.uni-kl.de/AG-Haerder/Ritter.html

Roller:
http://www.software.ibm.com/ad/flowmark/exmn0mst.htm

Rose:
http://www.faw.uni-ulm.de/

Salzberg:
http://www.ccs.neu.edu/home/salzberg/

Schek:
http://www-dbs.inf.ethz.ch

Schuster:
http://www6.informatik.uni-erlangen.de/Staff/schuster.html

Schwab:
http://www.buva.sowi.uni-bamberg.de/mitarbeiter/schwab.html

Unland:
http://www-wi.uni-muenster.de/pi/personal/index.htm
http://www.informatik.uni-essen.de/DW/

Vossen:
http://wwwmath.uni-muenster.de/~dbis/index.html
http://wwwmath.uni-muenster.de/~dbis/Vossen/index.html

Wanka:
http://www-wi.uni-muenster.de/pi/personal/wanka.htm

Weikum:
http://www-dbs.cs.uni-sb.de/

Weske:
http://wwwmath.uni-muenster.de/~dbis/Weske/index.html

Zuletzt erschienene und geplante Titel:

P. Cousot, R. Cousot, A. Mycroft (editors):
Abstract Interpretation, Dagstuhl-Seminar-Report; 123; 28.08.-01.09.95 (9535)

P. Brunet, D. Roller, J. Rossignac (editors):
CAD Tools for Products, Dagstuhl-Seminar-Report; 124; 04.09.-08.09.95 (9536)

C. Dwork, E.W. Mayr, F. Meyer a.d. Heide (editors):
Parallel and Distributed Algorithms, Dagstuhl-Seminar-Report; 125; 11.09.-15.09.95 (9537)

C. Hankin, H. R. Nielson (editors):
New Trends In the Integration of Paradigms, Dagstuhl-Seminar-Report; 126; 18.09.-22.09.95 (9538)

U. Herzog, G. Latouche, P. Tran-Gia, V. Ramaswami (editors):
Applied Stochastic Modelling in Telecommunication and Manufacturing Systems, Dagstuhl-Seminar-Report; 127; 25.09.-29.09.95 (9539)

L. Hordijk, G. Korn, A. Sydow (editors):
Modelling and Simulation of Complex Environmental Problems, Dagstuhl-Seminar-Report; 128; 02.10.-06.10.95 (9540)

J. André, A. Brüggemann-Klein, R. Furuta, V. Quint (editors):
Document Processing, Dagstuhl-Seminar-Report; 129; 16.10.-20.10.95 (9542)

J. Collado-Vides, R. Hofestädt, M. Löffler, M. Mavrovouniotis (editors):
Modelling and Simulation of Gene and Cell Regulation, Dagstuhl-Seminar-Report; 130; 23.10.-27.10.95 (9543)

F. Cucker, T. Lickteig, M. Shub (editors):
Real Computation and Complexity, Dagstuhl-Seminar-Report; 131; 06.11.-10.11.95 (9545)

K. Echtle, W. Görke, J.-C. Laprie, W. Schneeweiss (editors):
Quantitative Aspects of Designing and Validating Dependable Computing Systems -- Calculations, Measurements, and Simulations, Dagstuhl-Seminar-Report; 132; 13.11.-17.11.95 (9546)

J. Buchmann, R. Loos, R. Mäder (editors):
Computeralgebra - Software, Dagstuhl-Seminar-Report; 133; 05.02.-09.02.96 (9606)

O. Danvy, R.Glück, P. Thiemann (editors):
Partial Evaluation, Dagstuhl-Seminar-Report; 134; 12.02-16.02.96 (9607)

P.B. Andersen, M. Nadin, F. Nake (editors):
Informatics and Semiotics, Dagstuhl-Seminar-Report; 135; 19.02.-23.02.96 (9608)

S. Näher, H. Noltemeier, I. Munro (editors):
Data Structures, Dagstuhl-Seminar-Report; 136; 26.02.-01.03.96 (9609)

A. Bonner, A. Heuer, L. Tanca (editors):
New Trends in Database Languages, Dagstuhl-Seminar-Report; 137; 04.03.-08.03.96 (9610)

D. Dolev, R. Strong, R. Reischuk (editors):
Time Services, Dagstuhl-Seminar-Report; 138; 11.03.-15.03.96 (9611)

R. Bajcsy, R. Klette, W. Kropatsch, F. Solina (editors):
Theoretical Foundations of Computer Vision, Dagstuhl-Seminar-Report; 139; 18.-22.03.96 (9612)

V. Claus, J. Hopf, H.-P. Schwefel (editors):
Evolutionary Algorithms and their Application, Dagstuhl-Seminar-Report; 140; 25.-29.3.96 (9613)

U. Dayal, A. Kemper, G. Moerkotte, G. Weikum (editors):
Performance Enhancement in Object Bases, Dagstuhl-Seminar-Report; 141; 01.-04.04.96 (9614)

Ch. Lengauer, L. Thiele, M. Wolfe, H. Zima (editors):
Loop Parallelization, Dagstuhl-Seminar-Report; 142; 15.04.-19.04.96 (9616)

E. A. Lee, G. de Micheli, W. Rosenstiel, L. Thiele (editors):
Design Automation for Embedded Systems, Dagstuhl-Seminar-Report; 143; 22.-26.04.96 (9617)

M. Droste, E.-R. Olderog, B. Steffen, G. Winskel (editors):
Semantics of Concurrent Systems - Foundations and Applications, Dagstuhl-Seminar-Report; 144; 06.05.-10.05.96 (9619)

T. Nishizeki, R. Tamassia, D. Wagner (editors):
Graph Algorithms and Applications, Dagstuhl-Seminar-Report; 145; 13.05.-17.05.96 (9620)

M. Hanus, J. Lloyd, J. Moreno Navarro (editors):
Integration of Functional and Logic Languages, Dagstuhl-Seminar-Report; 146; 20.-24.5.96 (9621)

H. Bieri, G. Brunnett, T. DeRose, G. Farin (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 147; 27.05.-31.05.96 (9622)

P. Hanrahan, H. Müller, C. Puech (editors):
Rendering, Dagstuhl-Seminar-Report; 148; 10.06.-14.06.96 (9624)

A. Fiat, G. Woeginger (editors):
On-line Algorithms, Dagstuhl-Seminar-Report; 149; 24.06.-28.06.96 (9626)

J. Dix, D. Loveland, J. Minker, D. Warren (editors):
Disjunctive Logic Programming and Databases: Nonmonotonic Aspects, Dagstuhl-Seminar-Report; 150; 01.07.-05.07.96 (9627)

H.-D. Ehrig, F. von Henke, J. Meseguer, M. Wirsing (editors):
Specification and Semantics, Dagstuhl-Seminar-Report; 151; 08.07.-12.07.96 (9628)

F. Leymann, H.-J. Schek, G. Vossen (editors):
Transactional Workflows, Dagstuhl-Seminar-Report; 152; 15.07.-19.07.95 (9629)

W. Aspray, R. Keil-Slawik, D. Parnas (editors):
The History of Software Engineering, Dagstuhl-Seminar-Report; 153; 26.08.-30.08.96 (9635)

H. Bunke, R. Bolles, H. Noltemeier (editors):
Modelling and Planing for Sensor Based Intelligent Robot Systems, Dagstuhl-Seminar-Report; 154; 02.09.-06.09.96 (9636)

H. Ehrig, U. Montanari, G. Rozenberg, H.J. Schneider (editors):
Graph Transformations in Computer Science, Dagstuhl-Seminar-Report; 155; 9.-13.9.96 (9637)

U. Goltz, R. De Nicola, F. Vaandrager (editors):
Expressiveness in Concurrency, Dagstuhl-Seminar-Report; 156; 16.09.-20.09.96 (9638)

J. Bocca, H. Decker, A. Voronkov (editors):
Advances in Logic Databases, Dagstuhl-Seminar-Report; 157; 23.09.-27.09.96 (9639)

E. Allender, U. Schöning, K. Wagner (editors):
Structure and Complexityv, Dagstuhl-Seminar-Report; 158; 30.09.-04.10.96 (9640)

E. Nowak, J. Traub, G. Wasilkowski (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 159; 21.10.-25.10.96 (9643)

Ch. Gold, J. Snoeyink, F. Wagner (editors):
Computational Carthography, Dagstuhl-Seminar-Report; 160; 04.11.-08.11.96 (9645)

J. Hendler, J. Koehler (editors):
Control of Search in AI Planning, Dagstuhl-Seminar-Report; 161; 18.11.-22.11.96 (9647)

M. Broy, Ch. Floyd, J. Goguen, B. Paech (editors):
Formal Methods and Situated Cooperative Design in Software Development, Dagstuhl-Seminar-Report; 162; 25.11.-29.11.96 (9648)

G. Berry, W.P. de Roever, N. Halbwachs, A. Pnueli (editors):
Syncronous Languages, Dagstuhl-Seminar-Report; 163; 09.12.-13.12.96 (9650)