Engineering and Electronic Design Automation, Bournemouth/United Kingdom, April 7-8, 1994, S. 55-60

WORKFLOW MANAGEMENT IN SOFTWARE ENGINEERING PROJECTS

Andreas Oberweis
Universität Karlsruhe
Institut AIFB
D-76128 Karlsruhe
Germany
E-Mail: oberweis@aifb.uni-karlsruhe.de

Abstract

Large software engineering projects require computer support for collaborative development work. An efficient management of the flow of work items between different people or different groups of people is an important prerequisite for a successful software engineering project. Workflow management in a software engineering project must include planning and modelling of development activities, resource allocation, monitoring and control of activities, and support of collaborative work.

1 INTRODUCTION

Large software engineering projects require computer support for collaborative development work. Usually, a *software development process model* prescribes the structure of the development process, the synchronization aspects between different development activities, and the content of the deliverables (design documents, source codes, system documentation etc.) to be produced. Software development process models are defined at a *generic* level. There are several national standard models available such as SSADM, MERISE, V-MODELL, and models of organizations like IEEE, NASA, DOD, ESA. *Tailoring* to particular enterprise and project needs and allocation of activities to project team members lead to a project specific development process model.

An efficient management of the flow of work items between different people or different groups of people is an important prerequisite for a successful software engineering project. The main tasks of *workflow management* in a software engineering project can be summarized as follows:

- planning and modelling of development activities, based on a development process model.
- resource allocation.
- monitoring and control.
- support of collaborative work.

Specific problems arise from the fact that project teams and resources may be located at *geographically distributed* places.

This paper focuses on workflow modelling aspects. It is structured as follows: In Section 2, INCOME/STAR, an environment for the development of distributed information systems, is briefly surveyed. Section 3 describes several features of workflow management in INCOME/STAR. Section 4 compares our approach to related works and Section 5 concludes the paper with an outlook on future research topics.

2 THE INCOME/STAR ENVIRONMENT

In this paper we describe the workflow management concepts in INCOME/STAR which is a repository-based environment for the development of distributed information systems.

The development process of a distributed information system with INCOME/STAR (see Figure 1) includes database generation, application program generation, distribution of data and processes and a parallelization of processes. The development process is supported by prototyping based on the conceptual schema. High level Petri nets are used for the representation of the behaviour schema and a semantic object model for the representation of the data schema.

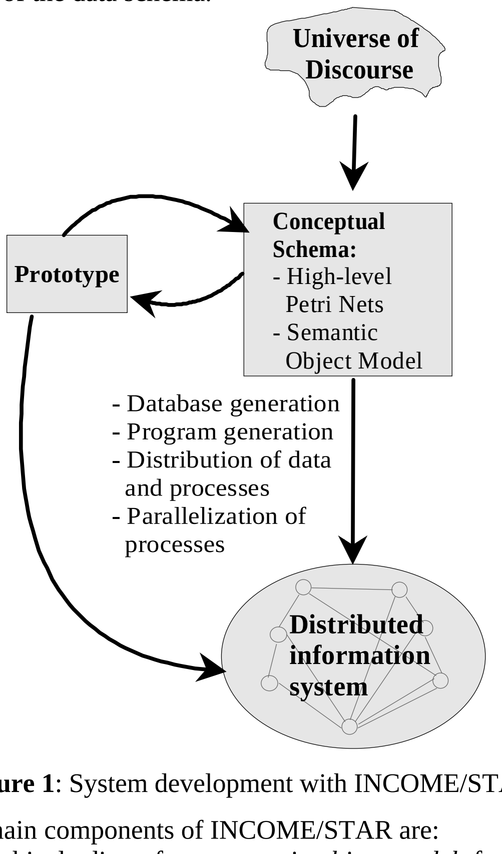

Figure 1: System development with INCOME/STAR

The main components of INCOME/STAR are:

- graphical editors for a *semantic object model*, for *high level Petri nets* and for other design documents (such as, e.g., *function hierarchies* and *object glossaries*).

- database and application program generators: the database generator produces a relational database definition (including forms and reports definitions) from the conceptual data schema. The application program generator produces C-Programs with embedded SQL-commands from the conceptual behaviour schema.
- simulation and prototyping facilities based on high-level Petri nets (Mochel et al. 1993).
- repository to store development documents and to maintain consistency of documents.
- communication facilities, extending electronic mail capabilities (Oberweis et al. 1994).
- electronic group calendar.
- workflow manager.

The INCOME/STAR repository supports shared access to the design documents and by this supports concurrent software engineering. System analysts, system developers, programmers and other project team members having access to the repository may be located at different sites in a network of workstations. Furthermore, the repository itself may be fragmented and geographically distributed.

The development process in INCOME/STAR is based on ProMISE, a **Pro**cess **M**odel for **I**nformation **S**ystem **E**volution (Scherrer et al. 1994).

3 WORKFLOW MANAGEMENT IN INCOME/STAR

3.1 Nested-Relation/Transition Nets (NR/T-Nets)

Petri nets[1] are a graphical language which allows to formally specify distributed system behaviour. *Sequences* of activities can be modelled as well as *conflict situations* between activities (mutual exclusion) and *concurrency*. INCOME/STAR uses a new type of high-level Petri nets, namely *Nested-Relation/Transition Nets* (*NR/T-nets*) (Oberweis and Sander 1992, Oberweis et al. 1993), as a formal basis for workflow management in software engineering processes.

To each place in an NR/T-net, a complex object type is assigned which is specified using the notation of the semantic hierarchy object model SHM (cf. (Brodie and Ridjanovic 1984)). Basic constructs for the structuring of data are *classification*, *aggregation*, *specialization* and *grouping*.

The marking of a place in an NR/T-net is a nested relation of the respective type, i.e. a set of so-called complex objects, where attribute values may again be nested relations. Typical deliverables in software development process models usually have a hierarchical structure which directly corresponds to a tuple in a nested relation.

A transition in an NR/T-net represents a class of operations on relations in the transition's input- and output-places, an occurrence of a transition denotes one single occurrence of the respective operation.

NR/T-nets are an upwards compatible extension of the well-known *predicate/transition nets* (Genrich 1987): the marking of a place in a predicate/transition net is given as a normalized relation where attribute values of a tuple are atomic, i.e. unstructured. This is obviously not appropriate for the modelling of operations on complex structured objects, since it does not allow, e.g., concurrent access to different set-valued attributes of the same complex object. An example is a situation where different development team members access different parts of the same development document.

3.2 Example

Figure 2 shows the graphical representation of the structuring concepts of SHM which is adopted from (Lausen and Schek 1987).

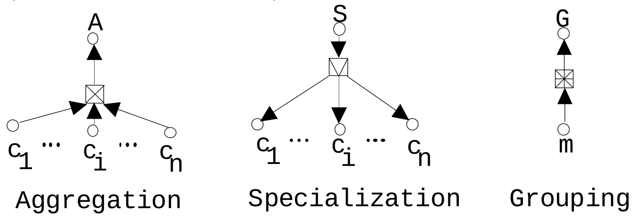

Figure 2: Structuring concepts in SHM

Figure 3 shows the structure of a (simplified) object type `DESIGN-DOCUMENT`.

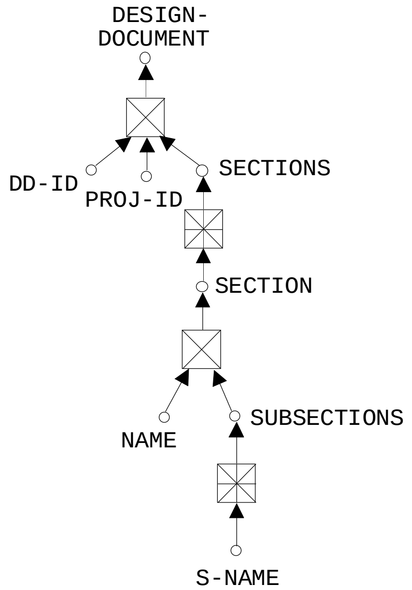

Figure 3: Type `DESIGN-DOCUMENT`

An object of type `DESIGN-DOCUMENT` is composed of a design document identifier (`DD-ID`), a project identifier (`PROJ-ID`), and a set of sections (`SECTIONS`). Each section (`SECTION`) is composed of a section name

[1] We suppose that the reader is familiar with the basic Petri net notation (see, e.g., (Genrich 1987)). Due to space limitations we cannot give a complete description here.

(NAME) and a set of subsections, where a subsection has a subsection name (S-NAME). DD-ID, PROJ-ID and S-NAME are atomic attributes.

Figure 4 shows the tabular representation of example documents doc1 and doc2 of project p1 and doc3 of project p2.

DESIGN-DOCUMENT			
DD-ID	PROJ-ID	SECTIONS	
		NAME	SUBSECTIONS
			S-NAME
doc1	p1	{<sec1,	{sn1,sn2,sn3}>,
		<sec2,	{sn1,sn2,sn3}>
		<sec3,	{sn1,sn2}> }
doc2	p1	{<sec1,	{sn1,sn2,sn3}>,
		<sec2,	{sn1}>}
doc3	p2	{<sec1,	{sn1,sn2}>,
		<sec2	{sn1,sn2,sn3}>,
		<sec3,	{sn1,sn2,sn3,sn4}>,
		<sec4,	{sn1,sn2,sn3}> }

Figure 4: Tabular representation of three example objects of type DESIGN-DOCUMENT

It is possible to model locking mechanisms for documents at different levels of granularity with NR/T-nets. Figure 5 shows an NR/T-net with three different transitions, each of them describing a different type of access to objects of the type DESIGN-DOCUMENT. A possible initial marking of the place DESIGN-DOCUMENT is shown in Figure 4.

Arcs in an NR/T-net are inscribed with so-called *filter tables* which select data to insert into the adjacent output-place or to remove from the adjacent input-place. Filter tables may be hierarchically structured in order to reflect the hierarchic structure of complex objects. This allows to access values which are located on the lower levels of the attribute hierarchy.

A transition is enabled for an instantiation of the variables in the filter tables assigned to the incoming and outgoing arcs iff:

- the respective (instantiated) tuples in the filter tables at the ingoing arcs are contained in the adjacent input places, and
- the respective tuples in the filter tables at the outgoing arcs are *not* contained in the adjacent output places, and
- the logical rule which is optionally inscribed to the transition is *true* for the given variable instantiation.

For the set valued attributes we distinguish between two cases:

So-called *closed* variables which are overlined, e.g. $\overline{X}$, always access complete attribute values. In Figure 5, $\overline{X}$ must be instantiated by complete sets of sections of a document. If, e.g., D is instantiated to doc1 and P to p1, then $\overline{X}$ must be instantiated to

```
{<sec1, {sn1,sn2,sn3}>,
 <sec2, {sn1,sn2,sn3}>
 <sec3, {sn1,sn2}>     }.
```

So called *open* variables, e.g. X in Figure 6, may be instantiated by an arbitrary subset of a set attribute value. If D is instantiated to doc1 and P to p1, then X may be instantiated, e.g., to

```
{<sec1, {sn1,sn2,sn3}>,
 <sec2, {sn1,sn2,sn3}> }.
```

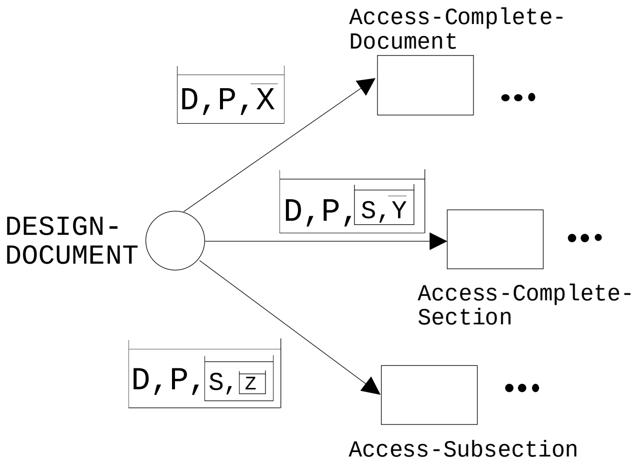

Figure 5: Example NR/T-net (A)

In Figure 5 the following different access types are modelled:

- When transition Access-Complete-Document occurs, it removes a complete document tuple from the input place DESIGN-DOCUMENT, e.g. the tuple
```
<doc1,p1,{<sec1, {sn1,sn2,sn3}>,
          <sec2, {sn1,sn2,sn3}>
          <sec3, {sn1,sn2}>      }>.
```
- When transition Access-Complete-Section occurs, it removes a single section of a given document tuple from the input place DESIGN-DOCUMENT. The transition may occur concurrently to itself with respect to different sections - possibly of the same document. This corresponds to a situation where different persons/tools access different sections of the same document at the same time.
- When transition Access-Subsection occurs, it removes a single subsection of a given section of a given design document from the input place DESIGN-DOCUMENT. The transition may occur concurrently to itself with respect to different subsections - possibly of the same section of the same document. This corresponds to a situation where different persons/tools access different subsections of the same document at the same time.

The meaning of the transitions in the NR/T-net given in Figure 6 is as follows:

- Access-Documents removes a subset X of sections of a document in place DESIGN-DOCUMENT, such that X contains the section <sec3,{sn1, sn2}>.
- Access-1-Subsection removes a subset Y of subsections of a document in place DESIGN-DOCUMENT, such that Y contains the subsection sn1.

Figure 7 shows an NR/T-net for the following situation: Two transitions insert subsections into design document

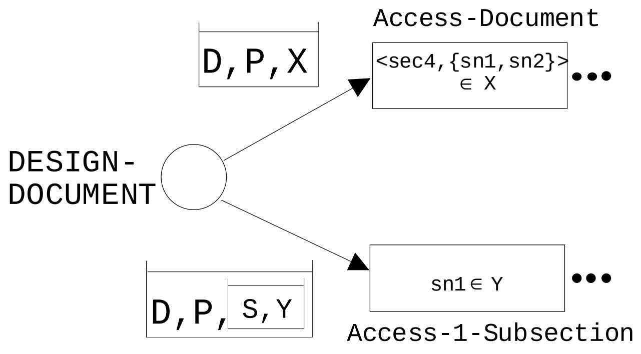

Figure 6: Example NR/T-net (B)

tuples in the place DESIGN-DOCUMENT. If the respective design document does not already exist in the place DESIGN-DOCUMENT then the corresponding tuple is inserted. Note that the variable names are local to each transition environment. D, S and Z have to be appropriately instantiated with respect to both transitions.

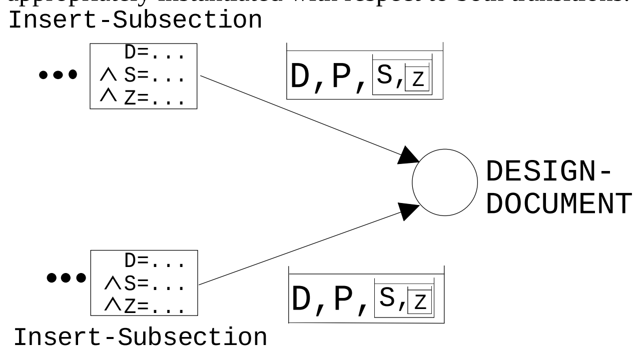

Figure 7: Example NR/T-net (C)

It is also possible to model access to more than one (sub-) tuple in a single filter table.

Figure 8 shows two activity classes as follows:

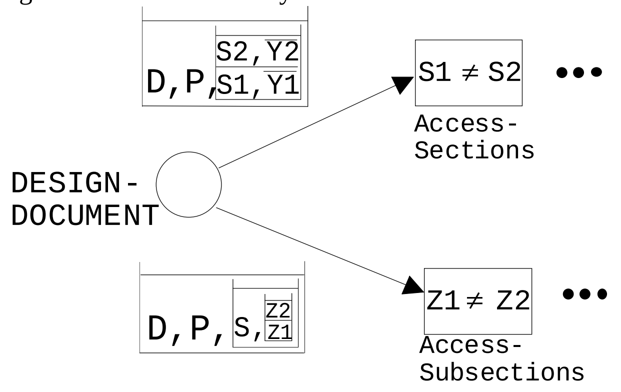

Figure 8: Example NR/T-net (D)

- When Access-Sections occurs, two different sections of the same design document are completely removed from the DESIGN-DOCUMENT place.
- When Access-Subsections occurs, two different subsections of the same section of the same design document are removed from the DESIGN-DOCUMENT place.

■

For a detailed description of NR/T-nets and formal definitions the interested reader is refered to (Oberweis and Sander 1992).

3.3 Workflow Modelling with NR/T-Nets

INCOME/STAR supports an evolutionary software development process model called ProMISE (Scherrer et al. 1994). The process model is specified at different levels of abstraction. At the deepest level it is directly executable by an available *Petri net simulator*, due to the formal semantics of our underlying net model. At the higher and less formal levels, it may be used as a graphical notation for communication between different groups of people involved in a software project.

The modelling of workflow in a software engineering project consists of two (complex) steps:

(i) ***Specification of the generic software engineering process model as a hierarchy of Petri nets***, where the bottom level nets are NR/T-nets and where the higher level nets are informal types of Petri nets, so-called *channel/agency nets*. NR/T-nets may include declarative constructs, namely so-called *fact transitions* (Oberweis and Sander 1993), to specify exceptional situations. Appropriate exception handling mechanisms can be specified in additional exception handling nets using the same formalism as for regular workflow.
The net may also include temporal knowledge, concerning e.g. clock times and calendar dates, duration of activities, deadlines, relative temporal relationships between activities (Oberweis and Lausen 1988).
The structure of the deliverables is specified in a semantic hierarchy object model.
Occasionally, this generic process model (including the descriptions of the deliverables) may be improved due to experiences made in earlier projects.

(ii) ***Instantiation of the generic model and tailoring due to the specific project needs***: Instantiating a workflow model given as an NR/T-net means to assign a start marking of the respective type to each place in the net. The marking of the places may denote, e.g., a certain resource allocation or an allocation of activities to project team members.
Tailoring means to delete certain parts of the generic model if certain activities or deliverables are not required in a software project. Existing activities may also be modified and certain new activities and deliverables may be introduced.

Finally, each NR/T-net represents a class of possible workflows in a software project. Simulation can be used to validate a given NR/T-net. Furthermore, it is possible to look at a certain workflow from different perspectives, e.g. to have a resource oriented view, a document oriented view, an activity oriented view, or a communication oriented view.

In INCOME/STAR workflow *modelling* is supported by three tools (see Figure 9): a workflow editor to edit workflows and the corresponding object structures, a workflow simulator to visualize and validate workflow models and a workflow analyzer for consistency checks.

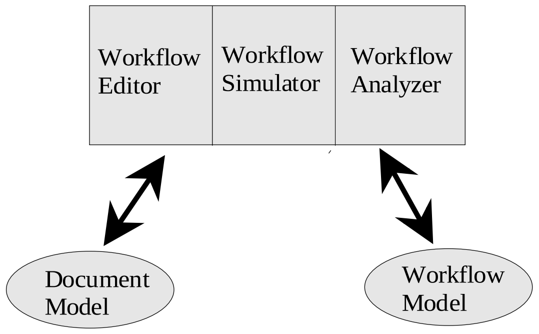

Figure 9: Workflow Modelling Support in INCOME/STAR

3.4 Workflow Management based on NR/T-Nets

The NR/T-net representation of the relevant workflows - including temporal aspects, resource allocation, personal responsibilities - is interpreted as a project plan, which is to be fulfilled for a concrete running project. The workflow manager in INCOME/STAR provides the following functionality:

- users are notified about actions that are to be started, completed or interrupted in a given situation.
- users are provided with specific information about the current system state.
- project managers are given an overview about the status of workflow.
- certain standard procedures are automated.
- activities and their outcome are controlled.
- alternative decisions, e.g. alternative resource allocations, may be validated.
- certain procedures may be simulated in advance, e.g. for training of unskilled persons or for analysis.
- users are provided with support for exception handling.

Central component of the workflow manager is the workflow engine (see Figure 10). The workflow engine is coupled to the repository containing all development documents and to the other components of the INCOME/STAR environment. The modification of values in the database or signals from other software systems trigger certain activities of the workflow engine. It is possible, e.g., to compare simulated values in the NR/T-net with actual values of the real system and to react appropriately if certain discrepancies occur.

4 RELATED WORK

In several other proposals different variants of Petri nets are used as a formal basis for the specification of workflow, e.g. the early work of (Zisman 1977) for office applications and, more recently, (Ellis and Nutt 1993) for general applications. However, both proposals do not integrate object related aspects, such as complex structured design documents. Our concept of NR/T-nets on the other hand integrates workflow and document modelling. It supports locking with different granularity and by this allows to model concurrent access to design documents at different levels.

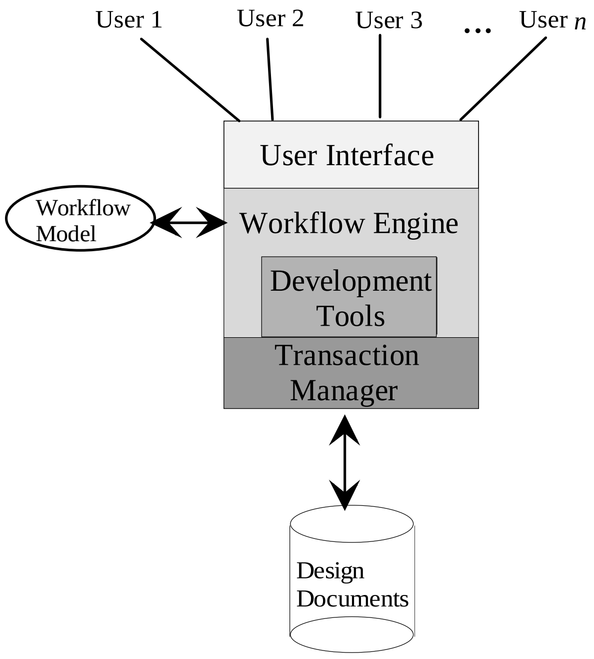

Figure 10: Architecture of the Workflow Manager in INCOME/STAR

(Gruhn and Jegelka 1992) propose another concept for software process modelling with Petri nets. An extension of high-level Petri nets, namely FUNSOFT nets, is used, which supports process analysis and process model enaction.

(Bussler and Jablonski 1993) describe general concepts for the modelling and execution of processes in workflow managment systems. They also include organizational aspects, e.g. authorization and roles, in their workflow model.

(Hahn et al. 1990) focus on social aspects of software projects and also consider aspects like problem negotiation, responsibilities for task fulfillment and task contraction in the area of software projects. Different models like so-called *group model for task cooperation*, *multi-agent conversation model for task-oriented negotiations* and *software process data model*, are proposed and integrated.

(Berztiss 1993) considers concurrent engineering of information systems. The software process model is represented in the executable SF (Set-Function) specification language, which also allows protoyping. However, it does not provide a graphical representation of workflows.

ConversationBuilder is a tool for collaborative software development described in (Kaplan et al. 1992). Based on

ConversationBuilder (Gintell et al. 1993) propose a collaborative inspection and review system for software engineering products. The system is tailorable to different development process models.

(Barghouti 1992) describes the cooperation facilities of MARVEL which is a rule-based software engineering environment. Rules are used to describe the development process model and to control the execution of the development tools.

(Pulli and Heikkinen 1993) propose concepts for concurrent engineering for real-time systems. They especially focus on aspects of prototyping which is combined with concurrent development and a special "redoing record-and-playback" mechanism.

(Godard 1993) discusses aspects of concurrency control related to cooperative system development from a more technical viewpoint.

5 OUTLOOK

INCOME/STAR is currently being implemented in a workstation environment. The user interface (including the graphical editors) is already realized in SMALLTALK, the simulation kernel in PROLOG (Mochel et al. 1993). A relational database system is used as basis for the repository.

An important aspect of our future research work concerns the development of a facility for an intelligent handling of exceptions during the software development process.

Another research direction concerns the replication of parts of the repository, which is not yet supported.

Acknowledgement

I wish to thank Gabriele Scherrer, Volker Sänger, Wolfgang Weitz, and Thomas Wendel for many useful comments on an earlier version of this paper.

REFERENCES

Barghouti, N.S. 1992. "Supporting cooperation in the MARVEL process-centered SDE." In: Proc. of the Fifth ACM SIGSOFT Symposium on Software Development Environments, H. Weber, ed. Software Engineering Notes, Vol. 17, No. 5, 21-31.

Berztiss, A.T. 1993. "Concurrent engineering of information systems." In: Information System Development Process, N. Prakash, C. Rolland, B. Pernici, eds. North-Holland, 311-324.

Brodie, M.L. and D. Ridjanovic. 1984. "On the design and specification of database transactions." In: On Conceptual Modelling, M.L. Brodie, J. Mylopoulos, J.W. Schmidt, eds. Springer-Verlag, 278-306.

Bussler, C. and S. Jablonski. 1993. "Process modeling and execution in workflow management systems." In: Proc. of the Third Annual Workshop on Information Technologies and Systems WITS'93, A.R. Hevner, N.N. Karmel, eds. Orlando/Florida, 152-161.

Ellis, C.A. and G.J. Nutt. 1993. "Modeling and enactment of workflow systems." In: Proceedings of the 14th Int. Conference on Application and Theory of Petri Nets 1993, M.A. Marsan, ed. Chicago, LNCS, Springer-Verlag, 1-16.

Genrich, H.J. 1987. "Predicate/transition nets". In: *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986*, W. Brauer, W. Reisig, G. Rozenberg, eds. *LNCS 254*. Springer-Verlag, 207-247.

Gintell, J. et al. 1993. "Scrutiny: A collaborative inspection and review system." In: Proc. Software Engineering - ESEC'93, I. Sommerville, M. Paul, eds. LNCS, Springer-Verlag, 344-360.

Godard, C. 1993. "COO: A transaction model to support Cooperating software developers COOrdination." In: Proc. Software Engineering - ESEC'93, I. Sommerville, M. Paul, eds. LNCS, Springer-Verlag, 361-379.

Gruhn, V. and R. Jegelka. 1992. "An evaluation of FUNSOFT nets." in: Proc. of the Second European Workshop on Software Process Technology, J.C. Derniame, ed. LNCS, Springer-Verlag, 1992.

Hahn, U.; M. Jarke; and T. Rose. 1990. "Group work in software projects." In: Multi-User Interfaces and Applications, S. Gibbs and A.A. Verrijn-Stuart, eds. North-Holland, 1990, 83-101.

Kaplan, S.M.; W.J. Tolone; A.M. Carroll; D.P. Bogia; and C. Bignoli. 1992. "Supporting collaborative software development with ConversationBuilder." In: Proc. of the Fifth ACM SIGSOFT Symposium on Software Development Environments, H. Weber, ed. Software Engineering Notes, Vol. 17, No. 5, 11-20.

Lausen, G.; H.J. Schek. 1987. "Semantic specification of complex objects." In: Proc. of the IEEE-CS Symposium on Office Automation, Gaithersburg/USA.

Mochel, T.; A. Oberweis; and V. Sänger. 1993. "INCOME/STAR: The Petri net simulation concepts." Systems Analysis - Modelling - Simulation, Journal of Modelling and Simulation in Systems Analysis, Vol. 13: 21-36.

Oberweis, A. and G. Lausen. 1988. "On the representation of temporal knowledge in office systems." In: Proc. AFCET/IFIP-TC8 Conference on Temporal Aspects in Information Systems (TAIS-87), C. Rolland, M. Leonard, F. Bodart, eds. Sophia-Antipolis/France, North-Holland, 131-145.

Oberweis, A. and P. Sander. 1992. "The specification of complex object behaviour by high-level Petri nets." Universität Karlsruhe, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Forschungsbericht 254, Karlsruhe.

Oberweis, A.; P. Sander; and W. Stucky. 1993. "Petri net based modelling of procedures in complex object database applications." In: Proc. Seventeenth Annual International Computer Software and Applications Conference COMPSAC'93, D. Cooke, ed. Phoenix/Arizona, 138-144.

Oberweis, A.; W. Stucky and T. Wendel. 1994. "Rechnergestützte Kommunikation in Software-Entwicklungsprojekten - Workgroup-Computing für kooperative Systementwicklung". In: Proceedings Online '94, 17. Europäische Congressmesse für Technische Kommunikation. Hamburg (in German, to appear).

Pulli, P.J. and M.P. Heikkinen. 1993. "Concurrent engineering for real-time systems." IEEE Software, 39-44.

Scherrer, G.; A. Oberweis and W. Stucky. 1994. "ProMISE - a process model for information system evolution." In: Proceedings Third Maghrebian Conference on Software Engineering and Artificial Intelligence, Rabat/Morocco (to appear).

Zisman, M.D. 1977. "Representation, specification and automation of office procedures." Ph.D. Thesis, University of Pennsylvania, Wharton School, Philadelphia.