INCOME/WF - A Petri Net Based Approach to Workflow Management

A. Oberweis*, R. Schätzle**, W. Stucky**, W. Weitz**, G. Zimmermann*

*Institut für Wirtschaftsinformatik
J.W. Goethe-Universität
D-60054 Frankfurt am Main
Germany
{oberweis|gzimmermann}@wiwi.uni-frankfurt.de

**Institut AIFB[1]
Universität Karlsruhe (TH)
D-76128 Karlsruhe
Germany
{schaetzle|stucky|weitz}@aifb.uni-karlsruhe.de

June 1996

Abstract

Flexibility of process support is a key requirement for current and future business applications. A technology which is designed to flexibly support various kinds of processes are *workflow management systems.* In this paper we present INCOME/WF - a prototype of a workflow management system based on high-level Petri nets as workflow modelling and execution language. Important requirements for INCOME/WF are easily adjustable workflow schemes and support for unstructured or weakly structured activities. Both requirements are not yet sufficiently fulfilled by existing WFMS.

1 Introduction

For almost thirty years database supported information systems have been used to implement numerous business application systems. Traditional database supported application systems in industry, business or public administration aim at the support of repetitive, non-creative activities. Recently, there is an increasing demand for systems which support unstructured or weakly structured creative "human-oriented" processes as well. These processes often require cooperation between different agents, and tools and methods to support them must be highly flexible as they should encourage human creativity rather than oppressing it by strict limitations. Business processes are embedded in a frequently changing environment which reacts sensitively to changing market conditions, the availability of new technologies, new laws or new management strategies. Therefore flexibility of processes is a key requirement for current and future business applications. A technology which is designed to support various kinds of processes are *workflow management systems (WFMS)* [ElN93, CoE95, GHS94, MWF92, She96, VoB96].

In this paper, we propose concepts for a flexible WFMS - INCOME/WF - which is based on high-level Petri nets. INCOME/WF[1] reuses many components of INCOME/STAR[2], which was implemented at the AIFB[3] Institute between 1991 and 1995. INCOME/STAR is a co-

[1] INCOME/WorkFlow
[2] STAR stands for "distribution" and "cooperation".
[3] Institut für Angewandte Informatik und Formale Beschreibungsverfahren (Institute for Applied Computer Science and Formal Description Methods)

operative environment for the development and maintenance of large, distributed information systems. The INCOME/WF- and the INCOME/STAR-project were partly supported by the Deutsche Forschungsgemeinschaft (DFG) in its program "Distributed Information Systems in Business". Both projects had the same general objective: support for distributed, cooperative processes. The first project - INCOME/STAR - took a more "traditional" approach based on conventional information system engineering methods and database technology, and mainly focused on software development. Although the INCOME/STAR project lead to valuable results (see below), it also confirmed what has already been stated above: Traditional, data-centered information systems have some deficiencies in supporting non-routine tasks which require a high amount of flexibility, such as the cooperative and creative subtasks of, e.g. software development. In a *software development environment* based on classic information systems, however, processes have to follow a rather strict scheme which is more or less "hard-wired" in the application systems built around a database.

Obviously, this problem is not limited to software development processes: Most business processes must also be easily adaptable to changing market factors, new technologies or strategic decisions. What is more, nearly every process in industry, business or public administration includes at least some creative and/or cooperative subtasks which should be easily adjustable to individual work styles and team dynamics. Hence, the follow-up project, INCOME/WF, deals with computer support for all kinds of distributed and cooperative processes, and uses modern information and communication technologies like WFMS instead of classic, data-centered information systems.

This paper is structured as follows: In Chapter 2, we briefly describe some basic concepts regarding WFMS. Chapter 3 summarizes the main results of the INCOME/STAR project and explains their impact on INCOME/WF. The central part of this paper, chapter 4, is devoted to INCOME/WF which is our concept of an advanced, Petri net based WFMS. Special emphasis is put on the subjects workflow modeling, architecture, support of unstructured or weakly structured workflows, and distributed execution of cooperative activities. Chapter 5 briefly surveys some related approaches, and the last chapter mentions some open problems and comments on possible future extensions.

2 Workflow management systems

2.1 Aims of WFMS

A *workflow* is a collection of activities organized to accomplish business processes like office or administration processes, shop-floor processes etc. [GHS94]. A WFMS "defines, manages and executes workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic".[4] The formal representation of a workflow is called *workflow scheme*. It describes the components of business processes: the activities to be performed, the corresponding data to be transformed and sequence relationships between single activities.

General aims of WFMS are:

- increase in productivity and reduction of costs,

[4] Glossary of the *Workflow Management Coalition*, available in the World Wide Web under URL http://www.aiai.ed.ac.uk/WfMC/DOCS/glossary.html

- increase in product quality and reduction of errors,
- improvement of customer satisfaction,
- higher flexibility concerning organizational changes.

Several information processing technologies have influenced the development of WFMS such as database management systems (active database management systems [BeM91, Cha89, Day88], object oriented database management systems [Heu92]) and groupware systems [Col92, EGR91].

2.2 Components of WFMS

As WFMS extend the capabilities of DBMS they have to provide further tools for the development of workflow schemes and the execution of workflows. The following list contains some components which are essential or at least desirable for a WFMS and briefly explains their purpose.

- Similar to DBMS, WFMS have to include a *dictionary*, but in a WFMS it does not only store information about the data scheme to be represented, like e.g. objects and relationships. Even more important is the information about the activities to be executed and the corresponding transformation of data.
- *Editors* for the modeling of workflow schemes and data schemes should be available. Every object to be transformed during the execution of a workflow must be modeled in the data scheme. Therefore controlled data exchange between different editors is necessary.
- *Simulators* can help to analyze, validate and improve workflows. Questions can be investigated like, e.g.
  - are the activities to be performed correctly specified?
  - can those activities be optimized?
  - are there any bottlenecks?

  As the simulator must execute the activities specified in the workflow scheme, it should have access to both the workflow scheme and the data scheme. Additionally, realistic test data should be available as a basis for system validation by simulation.
- During the execution of a workflow, the activities are controlled and driven by a so-called *workflow engine*. This component controls resources, triggers the activities which have to be performed next and informs persons involved in those activities.
- Different groups of persons involved in a workflow have to perform jobs, e.g. they have to produce or change documents. For this purpose *application programs* like text-processing systems, spreadsheet programs or document management systems have to be integrated. The WFMS must trigger those application programs and control the results. A monitoring component collects information about the current state of work, former and following system states and the sequence of activities.
- As already mentioned WFMS must support teamwork, because persons working with the system - possibly at geographically different locations - may have to interact during the execution of a workflow. For this reason, a WFMS should include a *groupware component*.

- Because users of WFMS are not necessarily computer professionals the system must provide various user interfaces for different types of users and applications. In many cases, *hypermedia* is an adequate interface for non-professional users.

3 INCOME/STAR - cooperative development and maintenance of distributed information systems

INCOME/STAR is an experimental, dictionary-based, cooperative software development environment which was designed and implemented at the AIFB Institute in Karlsruhe between 1991 and 1995 [OSS94]. Special emphasis was put on distributed, heterogeneous target systems like modern information system networks. The main components of the INCOME/STAR prototype are:

- graphical editors for *semantic object models*, for *high-level Petri nets* and for other design documents (such as, e.g., *function hierarchies* and *object glossaries*),
- simulation and prototyping facilities based on high-level Petri nets,
- database and application program generators,
- a dictionary to store development documents and to maintain consistency of documents,
- facilities for teamwork support and
- a component for software development process modeling and enactment.

In the following, a brief overview of the most important project results is given.

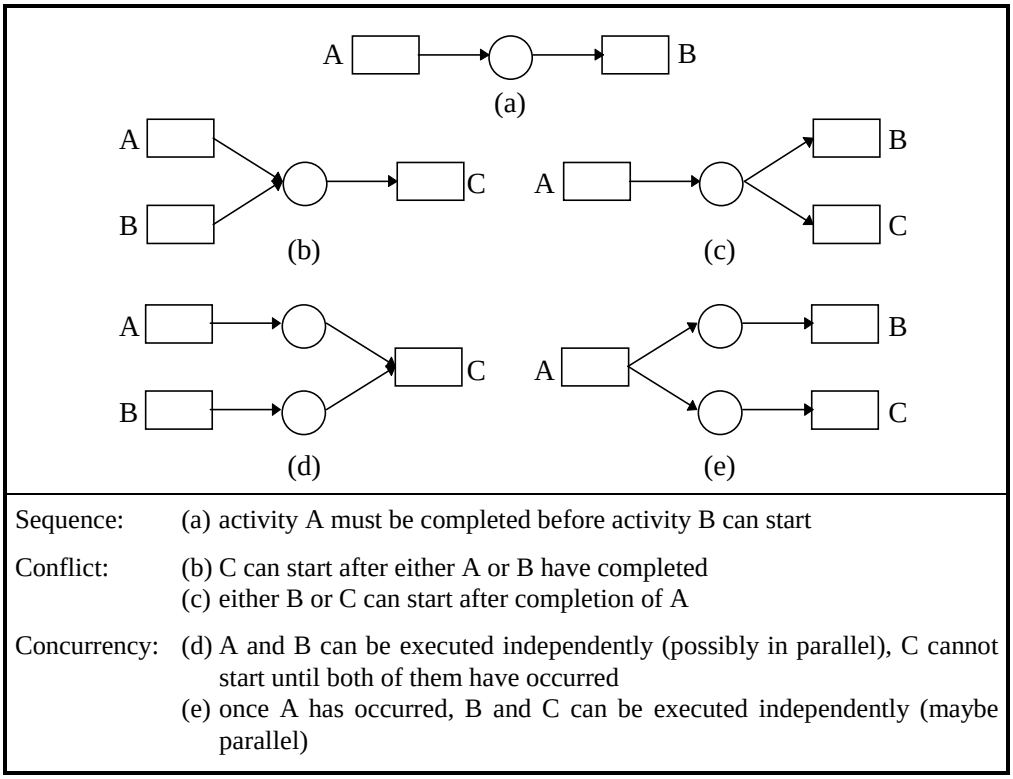
3.1 New methods for data and behavior modeling

Methods for conceptual modeling of information systems supported by INCOME/STAR combine high-level Petri nets (for behavior modeling) and semantic data models (for data modeling).

Petri nets are a graphical language for the formal specification of system behavior. A Petri net is a bipartite graph, consisting of two types of nodes, *places* and *transitions* which are connected by directed arcs. The usual graphical Petri net notation uses circles for places and rectangles for transitions (see Figure 1 for examples). Generally, a place represents an object storage, and a transition symbolizes an operation which removes objects (*tokens*) from its input places and inserts objects in the output places. An assignment of tokens to a place is denoted as *marking* of the place. The marking of all places at one point of time defines a global system state.

High-level Petri nets [BRR87] support an integrated description of object related and behavioral system aspects, because tokens in high-level Petri nets are structured objects with attributes to which values are assigned. *Predicate/transitions nets (Pr/T nets)* [Gen87], which are special high-level Petri nets, are closely related to relational databases: Places in a Pr/T net represent relation schemes in first normal form which may be marked with relations of the respective type. However, this type of net is too restrictive for adequately modeling the behavior of complex structured objects (i.e. objects consisting of other objects which may themselves be complex structured objects). Such objects cannot be assigned to a single place in a Pr/T net unless they are transformed into "flat" relations in first normal form. This transfor-

mation - in terms of the relational data model usually referred to as *normalization* - does not only produce a less "natural" representation of objects, it also restricts the expressiveness of the behavior scheme: concurrent access to different set-valued attributes of a complex structured object - which is a particular important behavior feature of cooperative information systems - cannot be modeled directly.



Sequence:	(a) activity A must be completed before activity B can start
Conflict:	(b) C can start after either A or B have completed (c) either B or C can start after completion of A
Concurrency:	(d) A and B can be executed independently (possibly in parallel), C cannot start until both of them have occurred (e) once A has occurred, B and C can be executed independently (maybe parallel)

Figure 1: Possible relationships between activities in Petri net notation

One of INCOME/STAR's most important results on the method side is the conception of *nested relation/transition nets (NR/T nets)* [Obe96a, ObS96], a novel type of high-level Petri nets for integrated modeling of concurrent processes and related complex structured objects in distributed applications. To each place in an NR/T net, a complex structured object type (a so-called *nested relation*) is assigned. A transition in an NR/T net represents a class of operations on relations in the transition's input- and output-places.

For the places in the nets, the local object structures are formally specified in a semantic data model. Since the object structures are to be mapped on nested relations, a data model which supports complex object structures is desirable. In INCOME/STAR, a model similar to the *semantic hierarchy model (SHM)* [BrR84] is used. Figure 2 shows the graphical representation of the basic data structuring concepts of the SHM: aggregation, specialization and grouping.

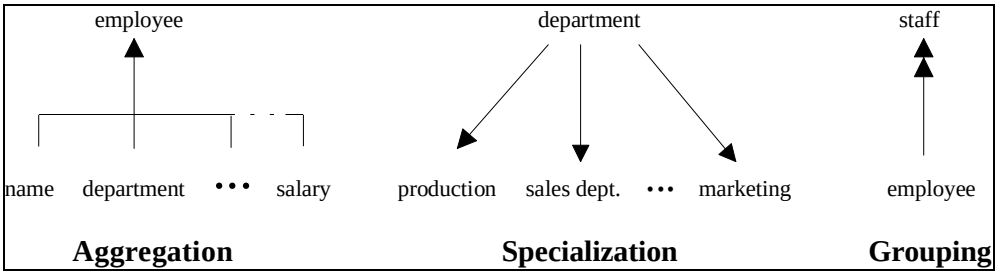


Figure 2: Structuring concepts in SHM

Although SHM and its structuring concepts fit quite well to nested relations and NR/T nets, the Entity Relationship model (ER model) is supported by INCOME/STAR as an additional alternative for data modeling, because of its widespread use and general acceptance. However, some problems arise when ER modeling is applied to the design of really large databases. There is, e.g., neither a way to obtain a general view nor to perceive the global context of a detailed data scheme with hundreds of entity and relationship types. Several approaches use ER model *clustering* to overcome these problems, e.g. [FeM86, RaS92, TWB89]. In extension of these approaches, INCOME/STAR offers three kinds of clustering:

Entity clustering creates an overview diagram from a detailed ER diagram by grouping related sections of the detailed diagram together into so-called entity clusters. These clusters represent (complex) entity types in the higher level ER diagram. The detailed relationship types between entity types within a cluster are omitted in the higher level ER diagram. The others - so-called outside-relationship types - are transformed to relationship types between the clusters which contain the original detailed entity types.

Simple relationship clustering was introduced to refine relationship types by several semantically similar ones. In this sense, simple relationship clustering is used top-down to formulate integrity constraints more precisely. It may also be applied bottom-up to cluster semantically similar relationship types into one single type.

Complex relationship clustering is proposed to refine relationship types by whole ER diagrams. In contrast to simple relationship clustering not only the relationship type is divided into several similar relationship types, but additional entity and relationship types may be introduced as well.

To combine the ER model and NR/T nets, a method was developed which maps hierarchically structured ER views onto nested relations. The ER view is derived from an ordinary ER scheme using SHM as an intermediate representation. A detailed description of INCOME/STAR's new concepts related to data modeling can be found in [Jae96].

3.2 Simulation concepts

In INCOME/STAR, simulation is an integral part of the system development process: simulators are connected to the central design dictionary, where the formal behavior specification is stored as a set of high-level Petri nets. Due to the formal semantics of the underlying net model, this specification is directly executable by a net interpreter and it may therefore be used for simulation. Hence, specifications can be verified at an early development stage and may be enhanced step by step in an evolutionary development process. INCOME/STAR provides an *open* simulation environment in a sense that application specific visualization modules may be integrated which can provide a problem oriented display of the current system state.

When dealing with large systems, multi-user support is required, because several developers are involved in the design process of the corresponding Petri net model. This includes

- access control for Petri net models to avoid inconsistencies when multiple developers try to manipulate a certain net at the same time,
- simultaneous visualization of a simulation run on an arbitrary number of workstations,
- a possibility for developers to influence a simulation run decentralized from their own workstation.

Simulation may either be manually controlled by users or driven automatically. One problem with automatically generated simulation runs is that they may consist of thousands of markings. Hence, it is not obvious how to check a given simulation run for certain behavior patterns which are of interest to a system designer. For this reason a novel graphical query language for simulation databases *GTL (**G**raphical **T**emporal **L**anguage)* [Sän96] was developed which combines capabilities of temporal and graphical database languages.

3.3 Process model support and cooperative system design

Software development with INCOME/STAR is driven by a software development process model called *ProMISE (**Pro**cess **M**odel for **I**nformation **S**ystem **E**volution)* [SOS94]. ProMISE describes the methodology supported by INCOME/STAR as well as organizational and cooperative aspects of the development process. Its main characteristics are evolutionary system development, incremental refinement of design documents, coupling to tools for specific methods and support of cooperative development.

Process model enactment offers active assistance to developers for tasks like monitoring development activities, document and workflow management, control of project responsibilities etc. The process model enactment system of INCOME/STAR consists of three parts:

- The first one, the ***model dictionary***, contains executable parts of a sample development process based on ProMISE ("process model fragments") *s*pecified in a Petri net based language named *PromiNets*. Prior to the execution of a fragment, the corresponding net must be *instantiated* with a project-specific marking, i.e. roles and team members, deliverables and deadlines etc. are associated with each other. This is done using the second component,
- the ***process editor***, which also allows an adaptation of a net to project specific needs ("tailoring"). Finally, the tailored and instantiated net model may be enacted by
- the ***process engine***, which uses the INCOME/STAR Petri net interpreter.

Process model fragments which are independent from each other may be processed in parallel, as the development process of large software systems is usually handled project teams. An efficient coordination of the development activities is an important prerequisite for a successful software engineering project. For this reason, the INCOME/ STAR environment contains a role-based groupware component called *RoCoMan (**Ro**le **Co**llaboration **Man**ager)* [Wen95].

3.4 Lessons learnt - from INCOME/STAR to INCOME/WF

A common problem with existing software development environments is the absence of support for (partly) unstructured, non-standard activities. Most of these activities are strongly human-oriented and include creative and cooperative tasks. A considerable amount of dynamics is inherent in such processes as they must be adaptable to individual human work styles and changing environmental conditions quite frequently. Hence, systems which support such tasks should provide enough flexibility.

The insight that software development is just one application area where flexible coordination and modeling of workflows plays a key role, lead us to the decision that workflow management could be an interesting research area by itself.

Moreover, WFMS and software development environments with active process support (like INCOME/STAR) have quite similar tasks [Fer93] as software development may be viewed as

a special kind of workflow. As a matter of fact, INCOME/WF can directly take up some of INCOME/STAR's results: First, Petri nets are a good instrument for integrated data and behavior modeling which is an important aspect of WFMS. Especially NR/T nets with their description facilities for distributed processes on complex structured objects may be a good basis for a workflow description language. Secondly, the existing Petri net interpreter can become the core of a workflow engine. Finally, some parts of INCOME/STAR's teamwork support system may be reused.

Of course, even with a broad base of knowledge and (partly) reusable prototypes to fall back on, our goal of building a prototype WFMS for flexible management of distributed, cooperative processes required some innovative concepts and the development of new methods and tools. These new aspects are summarized in the next chapter where we present INCOME/WF, our "vision" of an advanced WFMS.

4 INCOME/WF - workflow management based on Petri nets

In this chapter we survey some important aspects of the INCOME/WF prototype which is currently designed and implemented in a cooperation project between the AIFB Institute (Universität Karlsruhe) and the IWI[5] Institute (Universität Frankfurt/Main).

4.1 Workflow modeling

The INCOME/WF workflow modeling language [Obe96b] is based on high-level Petri nets. For several reasons we think that Petri nets are a particularly well-suited notation for workflow schemes:

- *Integration of data and behavior aspects,*
- *Support for concurrent, cooperative processes,*
- *Different degrees of formality,*
- *Availability of analysis techniques,*
- *Flexibility.*

Petri net schemes are directly executable by an interpreter (which is the workflow engine of the Petri net based WFMS), but they are not "hard-wired" application programs. This guarantees a reasonable degree of flexibility, because of the "late binding" between activities and objects. An adjustment of a workflow process is possible at run-time. As Petri net schemes can be changed easily, they are an ideal language for simulation and analysis of workflows in order to reorganize and improve workflows. Starting with reference processes for a certain business area, workflows can be tailored to the users' individual needs.

The advantages of Petri net based workflow modeling has been widely recognized. [Aal96] suggests a Petri net based conceptual standard for modeling and analysis of workflows and expects an impact on workflow modeling which is comparable to the impact the ER model or the relational data model had on DBMS.

[5] Institut für Wirtschaftsinformatik (Institute for Management Information Systems)

4.2 The architecture of INCOME/WF

Figure 3 surveys the architecture of INCOME/WF. The central component is the WFMS dictionary, a database which contains workflow data, i.e.

- data schemes describing the types of data items to be processed by the workflow system,
- workflow schemes, containing detailed descriptions of the workflows to be supported and
- information about currently active workflows (which are - possibly multiple - instances of their corresponding data/workflow schemes).

The dictionary also provides access to

- simulation data generated by the Petri net simulator,
- user data which may partly be organized in a database ("structured data") and partly consist of a more or less „unstructured“ collection of data files produced by various application programs,
- system data, such as passwords, user names, access control lists and so on.

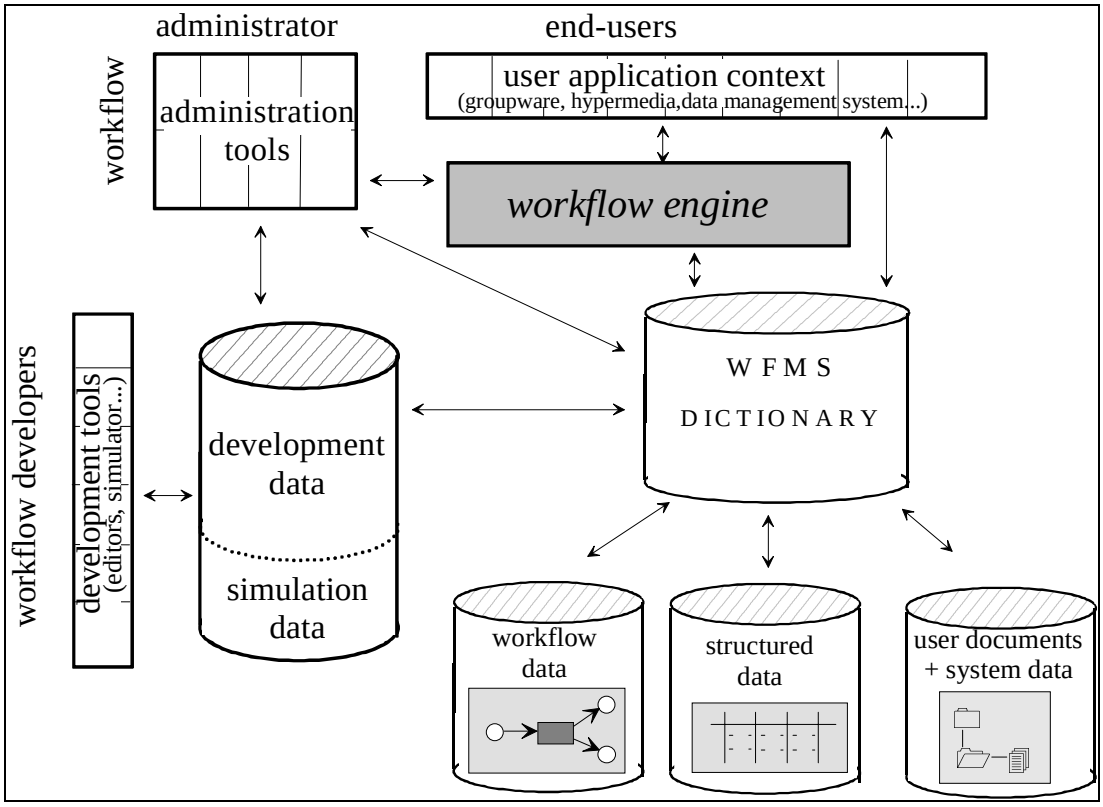


Figure 3: Architecture of INCOME/WF

The dictionary may be centralized or distributed, depending on the structure of the organization using the WFMS, technical capabilities (hardware and software) and the overall size of the WFMS. Distributed database management systems (DDBMS) provide a unified view of the complete data available in the system and thus enable workflow designers to access this information quickly and conveniently while at the same time the DDBMS may be running on many different nodes (computers). Parts of the dictionary may be stored locally (possibly replicated). Replication enhances data security and availability. On the other hand, additional administration effort is required for keeping consistency among the replicated data.

A user of the WFMS may play different roles which in turn determine his access privileges, user interface and many other attributes. The four major roles are:

- workflow system administrator,

- workflow developer,
- workflow administrator,
- end-user.

Depending on a user's role, the WFMS offers different applications and tools.

- ***Role: Workflow system administrator***

The role equipped with the most privileges is the one of *the workflow system administrator* since he has full access to all data in the system. His activities combine those of a system administrator on a traditional multi-user computer system, such as creating new user accounts and the installation / maintenance of application programs, and those of a database administrator, e.g. granting and revoking access privileges and optimizing the overall system performance. All these activities are supported by tools quite similar to the system administration programs for traditional multi-user operating systems and databases.

- ***Role: Workflow developer***

Workflow developers model and implement new workflow schemes using Petri nets. Special graphical editors are available to "draw" the model of the workflow, rather than to describe it in some sort of textual workflow programming language. Since models of complex workflows tend to become very large there is a need to allow multiple developers concurrent access to the same model. Hence the WFMS must enforce a locking scheme to prevent conflicting modifications by different developers.

Another important tool for developers is the workflow simulator which is used to validate a preliminary workflow scheme and to help choosing the best from a set of design alternatives. For later use of the simulation results, the sequence of system states reached during a simulation run may be recorded in a simulation database which can later be accessed by analysis tools. Simulator and editor are directly connected by a network link such that a developer who has just finished his work on a workflow scheme can send it directly from the editor to the simulator and can even watch an animation of the executing simulation in his editor window. When the workflow scheme is sufficiently tested and ready for "real" use, it is loaded into the workflow database where it can be accessed by the end-users.

- ***Role: Workflow administrator***

While a workflow system administrator is a person with comprehensive technical knowledge in the first place, a *workflow administrator* must have a good understanding of the problem domain of the workflows he has to manage. He supervises the executing instances of "his" workflows and supports end-users. For this purpose he uses graphical monitoring tools which visualize relevant information such as estimated completion times for certain activities, system load, number of documents waiting to be processed, and so on. If there are problems during the processing of a workflow, the workflow administrator may need to do some exception handling by manual intervention. The WFMS provides means to perform this in a consistency-preserving and safe way.

- ***Role: End-user***

In our terminology, an *end-user* is the person to be supported in his daily work by the WFMS. The WFMS helps to coordinate the work among different end-users and enables them to react quickly to new requirements. A new workflow may be initiated by an end-user with appropri-

ate access privileges at any point of time. From this starting point, the system guides the user through the necessary steps to create and modify the desired documents. It provides context sensitive online help wherever it is needed. When the work on a document is finished by one agent, the document is automatically forwarded to the next agent according to the procedures specified in the workflow scheme. In practice, there will be numerous situations where there are (possibly many) alternative stations for the processing of a given document. Since all admissible choices are included in the workflow scheme, the WFMS can make intelligent decisions to reach an optimal degree of "work load balancing" among the participating employees.

4.3 Support of unstructured or weakly structured workflows

Business processes and office work are partially unstructured, which means that work procedures do not always follow a rigid (and repeating) structure. They may consist of unstructured components like problem-solving, exception handling and communication with others. Even work procedures that seem to be routine tasks are interwoven with unstructured work [EIN87, Sac95, DiS93, Suc95]

[Sac95] describes two different views of office work: One is called the *explicit, organizational view*, where work is described in terms of business functions; this is the "official view" of work which is found in handbooks with organizational guidelines and rules and in organization charts. The other view is called the *activity-oriented view*. It describes the work that actually has to be done in order to reach certain goals. This knowledge mainly exists in the head of experienced employees and is usually not explicitly documented.

Nevertheless, most current WFMS mainly support structured and repeating tasks and their modeling tools only capture the explicit, organizational view of work. In order to support unstructured or weakly structured work, one has to use groupware systems. However, real work is neither purely structured nor purely unstructured but instead a constant change between both categories. So what is really needed is a tool that incorporates characteristics of WFMS and groupware systems.

4.3.1 Modeling

The first area where unstructured or weakly structured tasks are taken into consideration are the modeling capabilities of INCOME/WF. It is rarely necessary (aside from being hardly possible) to describe a workflow in full detail. The approach taken in INCOME/WF is to describe the so-called core workflows on a relatively abstract level. Core workflows are roughly the equivalent of business processes or office procedures. These procedures do not capture office work in it’s entirety but nevertheless form a good framework for its description. Instead of trying to specify the details at the same modeling phase, this task is left for a second modeling phase.

In this second modeling phase, which can take place when the workflow system is already in use, the individual end-users have the opportunity to describe the details, i.e. their so-called private workflows. Depending on the skill of the end-user, this can be done with the help of an experienced workflow developer or by the end-user himself.

The user can decide, if his (private) tasks - or part of these - really need workflow support (in the sense of a structured and well-defined task flow), or if he just wants support by some groupware concepts of INCOME/WF. In this way the end-user can adapt the system to his personal needs and his working style in a independent manner, i.e. he is not forced into the

role of an "assembly line blue collar worker in an office environment" [KoB95] as existing WFMS often do.

The longer the workflow application is in use, the more of it is utilized and the more it captures the structure of an organization. As the end-users can change and adapt their private workflows at any time, the system reflects the constantly changing nature of office work.

At the same time, this modeling approach solves another problem of conventional WFMS. Experience with the introduction of workflow-systems in organizations has shown, that sometimes WFMS do not lead to the desired flexibility but to a "cementation" of task flows. The reason behind this problem is the vast amount of time and effort that has to be invested using traditional workflow modeling and execution techniques, so that the willingness to change anything afterwards is quite low.

4.3.2 Support of cooperative work

Since it is not possible to capture the nature of unstructured or weakly structured work through rigid formalisms [WaW93], this sort of work can only be supported by providing an elaborate groupware infrastructure. This infrastructure has to support the exchange and the availability of information; structured information (e.g. in databases) as well as unstructured information (e.g. in operating system files) has to be included.

The INCOME/WF environment gives the user access to data stored in a relational database, as well as to documents which are administered by a document management system. The document management system is equipped with a security scheme, that is specifically designed for teamwork in order to overcome the deficiencies of security schemes [Bro93] found e.g. in operating systems.

An important aspect is the presentation of data on the user's desktop. There is a need to present all the necessary information but at the same time to avoid information overload. Here we introduced the concept of a *context* in INCOME/WF. People in offices usually have to work on different tasks. While they are working on a specific task they are working in a certain context, i.e. only specific information is necessary to get the work done and only a specific group of people is involved. Contexts differ in the required information and the involved people. This characteristic of work is exploited in the user-interface. The user is able to define his own working-contexts and to switch between them as needed. In this way the system can present relevant information in the current context.

In addition to information sharing communication also has to be addressed by a system that supports unstructured work. Unstructured work consists in large parts of communication between people, the creation and interaction of teams and strategic alliances, also known as social nets. The main communication tool in INCOME/WF is an E-mail system that supports structured messages similar to the Information Lens system [MGL93]. This is in contrast to the extended E-mail system of INCOME/STAR, where we used an approach based on the speech act theory [WiF86]. Experience has shown that this approach works well in certain application areas, but in general it makes too strong assumptions on how work is done and how communication takes place.

Another tool for communication and coordination is the notification tool. The user is able to show interest in certain events like the completion of a task, arrival of a document etc. and is henceforth informed whenever this event occurs. In combination with this notification

mechanism it is possible to formulate ECA-Rules (Event-Condition-Action rules), i.e. the user is able to tell the system that whenever a certain event occurs and an additional condition holds, the system should perform some pre-defined action. With the aid of this tool the user has the possibility to express weakly structured workflows. In this way we have a smooth transition between structured, weakly structured and unstructured activities.

Much work is done by groups instead of individuals. Groups are often created in an ad hoc manner. These groups (in the following called social nets) may have members from different hierarchical levels and even from different organizations. The social nets in which the individual user is involved are represented in INCOME/WF with the aid of a tool that shows the members of a social net (using icons) in a window on the user's desktop. Thereby the user may explicitly express his view of the organizational structure. Each social net is associated with a context and is accordingly displayed whenever the context is active. By clicking on an icon which represents a person, all information concerning that person can be reached. This includes the person's address, documents sent, documents received etc.; furthermore, e-mail can be sent directly, and for the future we have planned the integration of a video-conferencing system, so that face-to-face communication will be possible.

With this integrated combination of data-sharing, communication, coordination and representation of social nets INCOME/WF provides an infrastructure that supports all important aspects of unstructured and weakly structured cooperative work.

4.4 Distributed execution of cooperative activities

One advantage of using high-level Petri nets for workflow specification is that Petri nets allow formal analysis and are at the same time directly executable by a net interpreter. The common graphical notation for Petri nets can directly be used to visualize the workflow execution.

An obvious problem in this context is that most end-users of workflow applications do not have sufficient background knowledge to fully understand Petri net models. The main reason for this is that end-users usually cannot identify the relationship between the graphical net representation (i.e. rectangles, circles and interconnecting arcs) on the one hand and the terms of their respective application domain.

To bridge this gap, the INCOME/WF workflow engine offers a flexible and open environment for the execution of Petri nets, based on distributed object technology. It provides means to connect a graphical display of the whole or certain relevant parts of the executed system to a Petri net interpreter [SäW95]. A high-level graphics toolkit is used to ensure that the implementation of the visualization components itself can be done in a rapid-prototyping manner.

The workflow engine integrates different components:

- An information server keeping track of the distributed system components. Its main purpose is to help its clients locating required resources, such as some unloaded node running a workflow engine. In the future, aspects like global load balancing or system-wide logging and statistics may be integrated.
- Execution server programs which instantiate one or more Petri net interpreters on request. Each of these can independently execute a given Petri net workflow model and control a number of connected visualization components.
- Visualization programs which can be connected to a (running) workflow.

- External control components which can take control over running workflows.

With the exception of the information server which exists only once, an arbitrary number of these components may be installed on one or more workstations. Due to technical restrictions there can be only one instance of an execution server on a given machine (which is not a problem since an execution server may spawn as many independent workflow interpreter processes as are needed) and only one data keeping component of a given type (i.e. DBMS- or file-based).

Currently, most components are written in the object-oriented programming language Python [RoB91]. For the communication between the distributed parts of the systems the ILU infrastructure is used. This system, which has been developed at Xerox PARC, facilitates the creation of and method calls between distributed objects.

A user of the workflow system may request the workflow engine to be informed whenever some interesting event occurs. The following event types are available:

- an activity may occur (the respective transition in the workflow model is enabled),
- an activity occurs (the respective transition occurs),
- a data object is created (the respective object is inserted into a place),
- a data object is deleted (the respective object is deleted from a place).

This implies that the visualization application can be implemented very easily as an event-driven system that just reacts on the received events (cf. Figure 4). Since it is made sure that only relevant events are forwarded, the necessary 'administrative overhead' (event filtering, program flow control) can be kept very low. A short example may illustrate this: We want to implement a visualization program which should print out a line whenever a tuple is inserted into the data store '`jobs`' in interpreter `interpr`. The following code should give an idea of how easy the implementation of a simple visualization client is. For the sake of simplicity, we assume that the variable `interpr` already holds a proxy object for some running execution somewhere in the system which it may have queried from the information service or via an interactive execution selection box which is part of INCOME/WF:

```
import WFObserver
import OBS
import ilu

class Visualizer(WFObserver.WFObserver):
   def notify(self, execStepNo, eventType, objName, valueList):
         print 'a new job has arrived:', valueList[0]

obs = Visualizer(exec)
interpr.registerObserver(obs, 'jobs', OBS.Event.PlaceIns)

ilu.RunMainLoop()
```

The class `Visualizer` inherits the standard behavior from the class `WFObserver` in the previously imported module of the same name, which is part of INCOME/WF. The only thing a visualization client has to provide itself is a `notify` method that is called by the interpreter whenever necessary. This method is called with the current interpretation step number, the event type and name of the object (place/transition in the underlying Petri net workflow

scheme) the notification refers to and a list of values. The latter reflects the tuple which is inserted into a place or removed from a place, or the variable assignment under which a transition is enabled / has fired. The `registerObserver` method of the interpreter then attaches the instance of the `Visualizer` class, `obs`, to the interpretation and triggers the `notify` method whenever the event `PlaceIns` (insertion of a tuple into a place) for place `jobs` occurs.

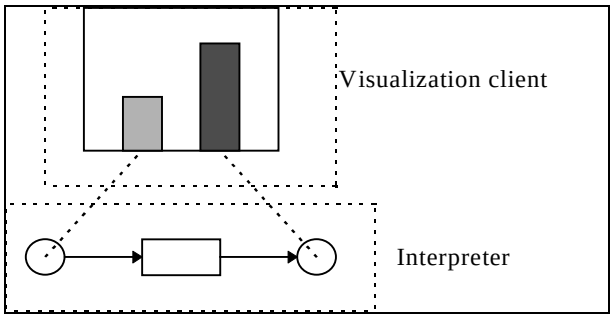


Figure 4: Interpreter and visualization clients

It should be noted that there is not necessarily a 1:1 relationship between interpreter and visualization clients. It is possible for a visualization client to register with different interpreters whenever this is helpful or to connect more than one visualization program to a running interpretation. Clients can even be dynamically added to and detached from an already executing workflow.

The great flexibility of Python makes it a natural choice not only to implement INCOME/WF components but also to serve as a scripting language to glue together and control distributed INCOME/WF services easily. With this purpose in mind, several Python operators have been overloaded to provide a natural and easy to understand notation as the following examples may show:

- Contact to a service running on some node in the net is simplified by instantiating an object of the desired type and providing the name of the service and the host name as parameters. Proxy objects are generated automatically. These objects automatically forward method calls to the real implementation of the object on the remote machine. In the following example, the variables '`db`' and '`inter`' are assigned proxy objects of a database service on host '`aifbknuth`' and a new interpreter instance spawned by an execution server on host '`aifbkorsar`', respectively:

```
db = PetriBase('PetriBase@aifbknuth.aifb.uni-karlsruhe.de')
inter = PetriInt('PetriInt@aifbkorsar.aifb.uni-karlsruhe.de')
```

- A database object is accessed like a two-dimensional associative array in Python where the first dimension specifies the type of data to be accessed (Petri net schemes or markings) and the second one the name of the desired net or marking. In the following example, the Petri net scheme with the name '`example1`' is retrieved from the remote database object and assigned to a variable `n`. After that, the marking '`mark1`' is copied back to the name '`copy1`' in the database:

```
n = db['net']['example1']
db['marking']['copy1'] = db['marking']['mark1']
```

- An interpreter object is seen as a special database object which can only hold one net and where the admissible names in the 'marking' dimension are restricted to those of the places occurring in the net. Continuing the example from above, the following line retrieves the Petri net scheme 'transport' from the database and loads it into the remote interpreter. The marking of two places in the net is fetched in the same way:

```
inter['net'] = db['net']['transport']
inter['marking']['station1'] = db['marking']['st1']
inter['marking']['station2'] = db['marking']['st88']
```

  All these operations are type-checked. An attempt to assign a marking to an incompatible scheme would lead to an exception, which can easily be handled using Python's built-in exception-handling constructs.

- Let int1 and int2 be interpreter instances and db be an instance of a database object. Then, the expressions

```
int1['marking']['warehouse1'] = int2['marking']['warehouse2']
                                + db['marking']['orders1']
```

  compute the set union of the marking of place 'warehouse2' from workflow int2 and the tuples in marking 'orders1' of the database object and marks place 'warehouse1' in interpreter int1 with the result. It should be noted that all three operands are objects that may be located on different nodes in the net, while syntactically the expression does not differ from a common addition/assignment where the operands are associative arrays. Implementation aspects of distribution and the different types of the operands (interpreter and a database objects) are completely transparent to the user.

- A final example: The interpretation in int1 has reached some interesting stage. We now wish to 'clone' the state of the (already running) interpreter int1 to some newly created interpreter int2 and have both execute synchronously as long as the marking of place 'customers' remains the same in int1 and int2:

```
int2 = PetriInt('PetriInt@aifbbrahms.aifb.uni-karlsruhe.de')
int2.copy(int1)

steps = 0
while int1['marking']['customers']==int2['marking']['customers']:
 int1.execStep()
 int2.execStep()
 steps = steps + 1

print "marking differs after", steps, "steps"
```

Registration of external components together with scripting provide very powerful means to implement arbitrary occurrence strategies for certain transitions, to interconnect different running workflows and to support "forward looking" simulation with Petri nets.

5 Related work

Several other approaches use Petri nets as a formal basis for the specification of workflows [ElN93] or business process modeling which was one driving force leading to the development of WFMS (as well as, on the other hand, WFMS are an enabling technology for business process modeling) [VoB96].

There are further proposals for extending Petri net based behavior modeling by semantic data modeling concepts, e.g. in the area of office automation, workflow management or information system design [EKT87, HPR93, Lau88, SoK93]. A common limitation of these approaches is that they do not offer sufficient modeling concepts for concurrent access to different components of the same complex structured object. This can lead to unnecessary locking of an object in a WFMS when different activities try to access different parts of this object at the same time.

A commercial WFMS using Petri nets is the *Leu* environment [DiG96]. Some other commercial WFMS are surveyed and discussed in [Jab95, Obe96a, HaL91]. A common drawback of many existing WFMS is, as it has been already stated above, a lack of support for unstructured or weakly structured activities.

6 Conclusion

In the area of information system development, behavior modeling has been recognized as being of similar importance as traditional data modeling. Consequently, there is a growing interest in WFMS supporting these two aspects appropriately. In our opinion currently available commercial WFMS mostly focus on features for behavior modeling and neglect aspects of data modeling.

In this paper a Petri net based approach was presented for integrated modeling of both data and behavior related aspects of workflow oriented business application systems. Since Petri nets have a well-defined formal semantics, various analysis and validation techniques can be applied to ensure a high level of quality for the resulting product. For the same reason, Petri nets are directly executable and thus can be used as an executable specification of the envisaged workflow, serving as direct input to the workflow engine.

Curently we are working on questions of workflow fragmentation and allocation in distributed environments, e.g. in Internet-based wide area networks. For database systems, database research has provided many elaborated concepts for data fragmentation, allocation and replication. However, the distributed management of workflows (i.e. of collections of business activities) is not yet sufficiently supported. There exist different (technical) proposals for distributed execution but there is no methodological support available for the planning and design of distributed workflow applications. We think that Petri nets are an appropriate model for fragmentation, allocation and replication of workflows, just as the relational data model is an appropriate model for fragmentation and allocation of data.

Acknowledgement

We wish to thank our former colleagues Peter Jaeschke, Volker Sänger, and Thomas Wendel for valuable discussions and many useful comments on earlier versions of this paper as well as for their contribution to INCOME/STAR and INCOME/WF.

References

[Aal96] W.M.P. van Aalst: Petri-net-based workflow management. In: [She96], pp.114-118

[BeM91] C. Beeri, T. Milo: A model for active object oriented databases. In: G.M. Lohmann, A. Sernadas, R. Camps (Eds.): Proc. 17th Int. Conf. on Very Large Data Bases, Barcelona (1991), pp. 337-349

[Bro93] J. Brooke: User interfaces for CSCW systems. In: [DiS93], pp. 23-30

[BrR84] M.L. Brodie D. Ridjanovic: On the design and specification of database transactions. In: M.L. Brodie, J. Mylopoulos, J.W. Schmidt (Eds.): On Conceptual Modelling, Springer (1984), pp. 278-306

[BRR87] W. Brauer, W. Reisig, G. Rozenberg (Eds): Petri Nets: Central Models and Their Properties. Advances in Petri Nets 1986, Springer (1987)

[Cha89] S. Chakravarthy: Rule management and evaluation: an active DBMS perspective. SIGMOD RECORD, Vol. 18 (3) (1989), pp. 20-28.

[CoE95] N. Comstock, C. Ellis (Eds.): Proc. Conf. on Organizational Computing Systems. Mipitas/California, ACM Press (1995)

[Col92] D. Coleman (Ed.): Groupware '92. Morgan Kaufman Publishers (1992)

[Day88] U. Dayal: Active database management systems. In: C. Beeri, J.W. Schmidt, U. Dayal (Eds.): Proc. 3rd Int. Conf. on Data and Knowledge Bases: Improving Usability and Responsiveness, Jerusalem (1988), pp. 150-169

[DiG96] G. Dinkhoff, V. Gruhn: Entwicklung Workflow-Management-geeigneter Software-Systeme. In: [VoB96], pp. 405-421

[DiS93] D. Diaper, C. Sanger (Eds.): CSCW in Practice: an Introduction and Case Studies. Springer (1993)

[EGR91] C.A. Ellis, S.J. Gibbs, G.L. Rein: Groupware: some issues and experiences. Communications of the ACM, Vol. 34 (1) (1991), pp. 39-58

[EKT87] J. Eder, G. Kappel, A.M. Tjoa, A.A.: Wagner: BIER - the behaviour integrated entity relationship approach. In: S. Spaccapietra (Ed.): Proc. 5th Intern. Conf. on the Entity-Relationship Approach, North-Holland (1987), pp. 147-168

[ElN87] C.A Ellis, N. Naffah: Design of Office Information Systems. Springer (1987)

[ElN93] C.A. Ellis, G.J. Nutt: Modeling and enactment of workflow systems. In: M.A. Marsan (Ed.): Proc. 14th Int. Conf. on Application and Theory of Petri Nets, Chicago, Springer (1993), pp. 1-16

[FeM86] P. Feldman, D. Miller: Entity model clustering: structuring a data model by abstraction. The Computer Journal, Vol. 29 (4) (1986), pp. 348-360

[Fer93] C. Fernström: PROCESS WEAVER: adding process support to UNIX. In: Continuous Software Process Improvement, Proc. 2nd Int. Conf. on the Software Process, Berlin, IEEE Computer Society Press (1993), pp. 12-26.

[Gen87] H.J. Genrich: Predicate/transition nets. In: [BRR87], pp. 207-247

[GHS94] D. Georgakopoulos, M. Hornick, A. Sheth: An overview of workflow management: from process modeling to workflow automation infrastructure. In: Distributed and Parallel Databases, Kluwer Academic Publishers (1994)

[HaL91] K. Hales, L. Lavery: Workflow Management Software: the Business Opportunity. Ovum Ltd. (1991)

[Heu92] A. Heuer: Objektorientierte Datenbanken. Addison-Wesley (1992)

[HPR93] C.A. Heuser, E.M. Peres, G. Richter: Towards a complete conceptual model: Petri nets and Entity-Relationship diagrams. Information Systems, Vol. 18 (5) (1993), pp. 275-298

[Jab95] S. Jablonski: Workflow-Management-Systeme: Modellierung und Architektur. International Thomson Publishing (1995)

[Jae96] P. Jaeschke: Integrierte Unternehmensmodellierung - Techniken zur Informations- und Geschäftsprozeßmodellierung. Deutscher Universitäts-Verlag (1996)

[KoB95] S. Koshafian, M. Buckiewicz: Introduction to Groupware, Workflow and Workgroup Computing. John Wiley & Sons (1995)

[Lau88] G. Lausen: Modelling and analysis of the bahaviour of information systems. IEEE Transactions on Software Engineering, Vol. 14 (11) (1988), pp. 1610-1620

[MGL93] T.W. Malone, K.R. Grant, K. Lai, R. Rao, D.A. Rosenblitt: The information lens, an intelligent system for information sharing and coordination. In: R.M. Baecker (Ed.): Readings in Groupware and Computer-Supported Cooperative Work. Assisting Human-Human Collaboration, Morgan Kaufmann Publishers (1993), pp. 461-473

[MWF92] R. Medina-Mora, T. Winograd, R. Flores, F. Flores: The action workflow approach to workflow management technology. In: J. Turner, R. Kraut (Eds.): CSCW´92 Sharing Perspectives, Proc. Conf. on Computer Supported Cooperative Work, Toronto, ACM Press (1992), pp. 281-288

[Obe96a] A. Oberweis: Verteilte betriebliche Abläufe und komplexe Objektstrukturen. Ein integriertes Modellierungskonzept für Workflow-Managementsysteme. B.G. Teubner (1996)

[Obe96b] A. Oberweis: An integrated approach for the specification of processes and related complex structured objects in business applications. Decision Support Systems, Vol. 17 (1996), pp. 31-53

[ObS96] A. Oberweis, P. Sander: Information system behavior specification by high-level Petri nets. To appear in: ACM Transactions on Information Systems (1996)

[OSS94] A. Oberweis, G. Scherrer, W. Stucky: INCOME/STAR: Methodology and tools for the development of distributed information systems. Information Systems, Vol. 19 (8) (1994), pp. 641-658

[RaS92] O. Rauh, E. Stickel: Entity tree clustering - a method for simplifying ER design. Proc. 11th Int. Conf. on the Entity-Relationship Approach, Karlsruhe (1992), pp. 62-78

[RoB91] G. van Rossum, J. de Boer: Interactively testing remote servers using the Python programming language. CWI Quarterly, Vol. 4 (4) (1991), pp. 283-303.

[Sac95] P. Sachs: Transforming work: collaboration, learning and design. Communications of the ACM, Vol. 38 (9) (1995), pp. 36-44

[Sän96] V. Sänger: Eine grafische Anfragesprache für temporale Datenbanken. Shaker (1996)

[SäW95] V. Sänger, W. Weitz: Entwurfsvalidation für verteilte Informationssysteme mit dem graphischen, mehrbenutzerfähigen Pr/T-Netz-Simulator GAPS+. In: F. Huber-Wäschle, H. Schauer, P. Widmayer (Eds.): GISI 95 - Herausforderungen eines globalen Informationsverbundes für die Informatik, Springer (1995), pp. 407-414

[She96] A. Sheth (Ed.): Proc. NSF Workshop on Workflow and Business Process Automation in Information Systems: State-of-the-art and Future Directions. Athens/Georgia, (1996)

[SoK93] A. Solvberg, D.C. Kung: Information Systems Engineering. Springer (1993)

[SOS94] G. Scherrer, A. Oberweis, W. Stucky: ProMISE - a process model for information system evolution. Proc. 3rd Maghrebian Conf. on Software Engineering and Artificial Intelligence, Rabat (1994), pp. 27-36

[Suc95] L. Suchman: Making work visible. Communications of the ACM, Vol. 38 (9) (1995), pp. 56-65

[TWB89] T.J. Teorey, G. Wei, D.L. Bolton, J.A. Koenig: ER model clustering as an aid for user communication and documentation in database design. Communications of the ACM, Vol. 32 (8) (1989), pp. 975-987

[VoB96] G. Vossen, J. Becker (Eds.): Geschäftsprozessmodellierung und Workflow-Management - Modelle, Methoden, Werkzeuge. International Thomson Publishing (1996)

[WaW93] D.G. Wastell, P. White: Using process technology to support cooperative work: prospects and design issues. In: [DiS93], pp. 105-126

[Wen95] T. Wendel: Computerunterstützte Teamarbeit in einer verteilten Software-Entwicklungsumgebung. Deutscher Universitäts-Verlag (1995)

[WiF86] T. Winograd, F. Flores: Understanding Computers and Cognition. Ablex (1986)