INCOME/STAR: Facing the Challenges for Cooperative Information System Development Environments

Andreas Oberweis[1], Wolffried Stucky[2], Gabriele Zimmermann[1]

Abstract

This paper surveys some innovative features of INCOME/STAR, an experimental environment for cooperative development of information systems.

First an extension of high-level Petri nets is described: *NR/T-nets* allow modeling of concurrent processes and related complex structured objects in distributed business applications.

Further new concepts have been developed for *entity* and *relationship clustering* to support a stepwise top-down approach for entity/relationship based object modeling. Distributed *multi-user simulation and prototyping* are proposed for the evaluation and analysis of NR/T-nets and the involved object schemata.

Then, *ProMISE* - an evolutionary *process model* for information system development - is surveyed. A role-based *groupware component* is part of the INCOME/STAR architecture to support communication, organization and social interaction in development projects.

Keywords

cooperative system design, Petri nets, information systems, Petri net simulation, software development environments, software process support

1 Introduction

Beyond all doubt information ranks as one of the most essential resources in industry, business and administration - it is at least as important as monetary budget, raw materials, personnel or machines. Hence, a major part of today's software systems are large, database supported information systems.

Providing environments which support an efficient production of high-quality information systems has been - and still is - a major objective of information systems engineering. But an industrialization of software engi-

[1] Institut für Wirtschaftsinformatik II, J.W. Goethe-Universität, 60054 Frankfurt/Main

[2] Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe (TH), 76128 Karlsruhe

neering in general and information systems engineering in particular is far from being achieved. More than a decade after the invention of CASE technology, there are still exciting challenges in this area, mainly because information systems have advanced in many aspects:

Complexity: Information systems are not only supposed to be specially suited for a certain application domain but also to support a wide range of functionality within this domain. On the one hand they must be capable, e.g., to handle production control data, on the other hand they must manage business and administration data of an enterprise.

***Distribution**:* Systems may be geographically distributed and are quite frequently integrated in networks.

***Interoperability**:* Systems are supposed to communicate and exchange data with other systems.

***Flexibility**:* Requirements change frequently due to market factors, new technologies or strategical decisions. Moreover, systems are embedded in a heterogeneous software or hardware configuration which is subject to change.

The objective of the INCOME/STAR project[3] is to detect deficiencies of existing development support systems and to implement a prototype of an integrated environment supporting cooperative development of large, distributed information systems in the above sense. The INCOME/STAR prototype is based on INCOME (**I**nteractive **N**et-based **Co**nceptual **M**odeling **E**nvironment), an existing tool for conceptual modeling and prototyping of information systems. INCOME was originally developed at our institute between 1985 and 1990. The main concepts of this academic version of INCOME are: integration of structural and behavioral system aspects, prototyping facilities and design dictionary support (cf. Lausen et al. 1989). Based on these concepts, a commercially available methods and tools package was developed. The commercial product INCOME (cf. INCOME 1994) is embedded in the ORACLE*CASE product family. ORACLE*CASE supports CASE*Method, an information engineering approach for database oriented system development (cf. Barker 1990). INCOME extends the ORACLE*CASE environment by providing modeling facilities for behavioral aspects of the target system, such as business process modeling, exception handling and temporal restrictions.

[3] The INCOME/STAR project is partially supported by the Deutsche Forschungsgemeinschaft DFG under grant Stu 98/9 in the program "Distributed Information Systems in Business".

While INCOME is primarily suited for the development of new information systems, INCOME/STAR supports both the development of completely new systems and the integration of new components into existing hardware and software environments. Special emphasis is put on distributed, heterogeneous target systems (like modern information system networks).
These target systems require new or adapted methods and advanced simulation and prototyping concepts. Software process support and cooperative design techniques were identified as additional important research fields in the context of system development.
This paper summarizes the new concepts of INCOME/STAR which can be grouped into four research directions: methodological extensions (Section 2), software process support (Section 3), advanced simulation and prototyping concepts (Section 4) and cooperative system design (Section 5). Section 6 mentions some related approaches while the final section documents first practical experiences and gives an outlook on future research work.

2 Methodological Extensions

Modeling highly flexible information systems requires object structures that are not as restricted as postulated by the relational data model. Methodological extensions of INCOME/STAR aim at providing concepts for a behavior and structure model which is adequate for complex structured objects. An important step towards this goal is the conception of NR/T-nets- a new variant of high-level Petri nets closely related to NF^2 (Non First Normal Form) relational databases (cf. Schek, Scholl 1986) - for behavior modeling (Section 2.1).
Another useful approach to cope with complex data and process structures is the use of hierarchically structured models which allow an incremental approach to conceptual modeling. Refinement and coarsening of Petri nets has already proven a successful technique in INCOME. INCOME/STAR provides an equivalent concept on the data side which extends existing Entity-Relationship model clustering techniques (Section 2.2).

2.1 Nested Relation/Transition Nets

Petri nets[4] are a graphical language for the formal specification of distributed system behavior. INCOME/STAR uses a new type of high-level Petri

[4] We suppose that the reader is familiar with the basic Petri net notation (cf. e.g. Reisig 1985).

nets, called nested relation/transition nets (NR/T-nets) (cf. Oberweis, Sander, Stucky 1993).
To each place in an NR/T-net, a complex structured object type is assigned, specified in a semantic data model similar to SHM (Semantic Hierarchy Model) (cf. Brodie, Ridjanovic 1984). Basic constructs for data structuring are classification, aggregation, specialization and grouping. Figure 1 shows the graphical representation of these concepts.

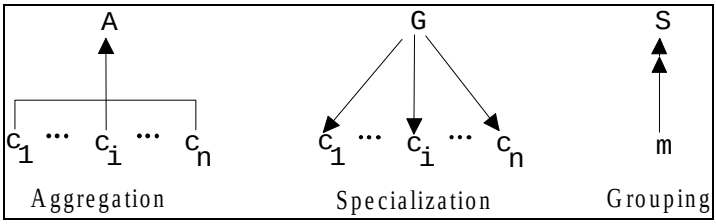

Fig. 1: Structuring concepts in SHM

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.
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. Operations may not only operate on whole tuples of a given relation but also on 'subtuples' of existing tuples.
NR/T-nets are an upwards compatible extension of the well-known predicate/transition nets (Pr/T-nets) (cf. Genrich, Lautenbach 1981): the marking of a place in a Pr/T-net is given as a normalized relation where attribute values of a tuple are atomic, i.e. unstructured. This is obviously not appropriate for modeling operations on complex structured objects, since it does not allow, e.g., concurrent accesses to different set-valued attributes of the same complex object. An example is a situation where different project team members access different parts of the same document.

Example
Figure 2 shows the structure of a (simplified) object type `DOCUMENT`.

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

of subsections (SUBSECTIONS). D-ID, PROJ-ID, NAME and SUBSECTION are atomic attributes.

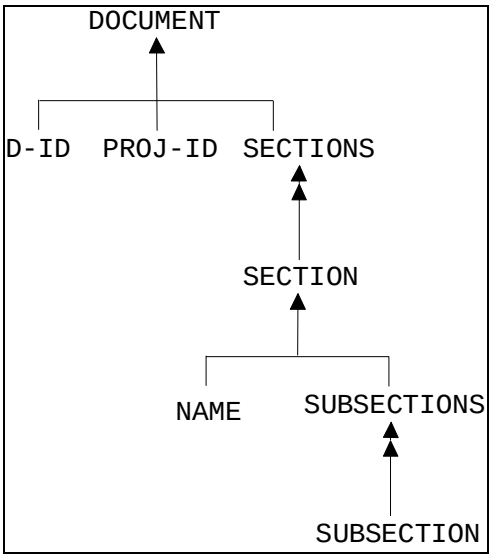

Fig. 2: Type DOCUMENT

Figure 3 shows the tabular representation of three example documents, doc1 and doc2 of project p1 and doc3 of project p2.

DOCUMENT			
D-ID	PROJ-ID	SECTIONS	
		NAME	SUBSECTIONS
			SUBSECTION
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}> }

Fig. 3: Tabular representation of three example objects of type DOCUMENT

Figure 4 shows an NR/T-net with three different transitions, each of them describing a different type of access to objects of the type DOCUMENT. A possible initial marking of the place DOCUMENT is given in Figure 3.

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

For the set valued attributes we distinguish between two cases:
So-called *closed* variables which are overlined, e.g. $\overline{X}$ in Figure 4, always access complete attribute values. In Figure 4, $\overline{X}$ 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}>     }.
```

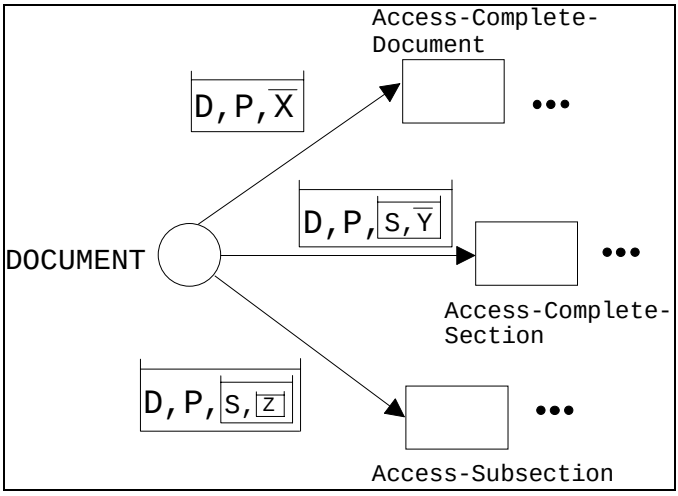

Fig. 4: Example NR/T-net (A)

So-called *open* variables, e.g. X in Figure 5, 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}> }.
```

In Figure 4 the following different access types are modeled:

- When transition `Access-Complete-Document` occurs, it removes a complete document tuple from the input place `DOCUMENT`, e.g.

```
<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 `DOCUMENT`. The transition may occur concurrently to itself or to other transitions 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 `DOCUMENT`. The transition may occur concurrently to itself or to other transitions 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 in Figure 5 is as follows:

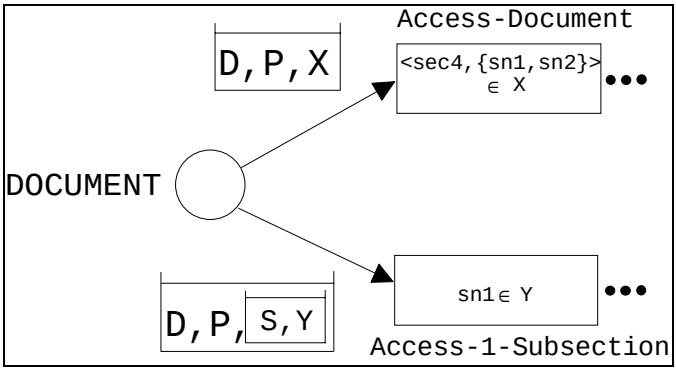

Fig. 5: Example NR/T-net (B)

- `Access-Document` removes a subset `X` of sections of a document in place `DOCUMENT`, such that `X` contains the section `<sec4, {sn1,sn2}>.`

- `Access-1-Subsection` removes a subset `Y` of subsections of a document in place `DOCUMENT`, such that `Y` contains the subsection `sn1`.

For a detailed description of NR/T nets and further examples, the interested reader is referred to (Oberweis, Sander, Stucky 1993).

2.2 ER Model Clustering

The Entity-Relationship approach is a widely accepted method for conceptual database design. However, some problems arise when ER modeling is applied to the design of really large databases concerning whole enterprises. There is no way to obtain a general view or to perceive the global context of a detailed enterprise schema with hundreds of entity and relationship types.
Several approaches use ER model clustering to overcome these problems (cf. Feldman, Miller 1986; Teorey et al. 1989; Rauh, Stickel 1992). Sections of the detailed diagram are mapped into so-called entity clusters, which are presented as (complex) entity types in a higher level ER diagram. All approaches are based on an already existing detailed ER diagram. Based on this, the abstraction layers are built *bottom-up*.

INCOME/STAR extends the approaches described above. It distinguishes between three kinds of clustering (cf. Jaeschke, Oberweis, Stucky 1993):
① *Entity clustering* was first proposed in (Feldman, Miller 1986). An overview diagram leaving out several details is created from a detailed ER diagram. Whole sections of the detailed diagram are collected into so-called entity clusters, which are represented as (complex) entity types in a higher level ER diagram. The detailed relationship types between entity types existing in one cluster are disappearing in the higher level ER diagram. The others - so-called outside-relationship types - are transformed to relationship types between the clusters containing the originally detailed entity types. The higher level diagram is abstracted iteratively by this method.

② *Simple relationship clustering* is newly introduced to refine relationship types by several semantically similar ones. *Simple relationship clustering* is used to formulate integrity constraints more precisely. In Figure 6, the relationship type 'works at' is refined by simple relationship clustering in the context of the refinement of 'Employee'. It is expressed that only members of the ground staff work at airports and that each member of the ground

staff works at *exactly one* airport. *Simple relationship clustering* can also be applied to represent integrity constraints in an ER diagram and to cluster semantically similar relationship types into one

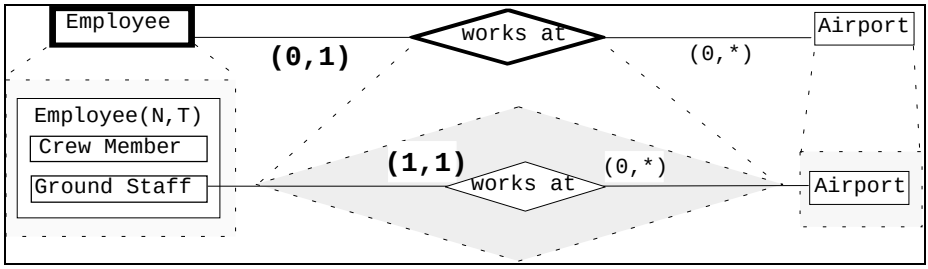

Fig. 6: Refinement of 'Employee' based on entity clustering; refinement of 'works at' based on simple relationship clustering

③ *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, additional entity and relationship types are introduced as well. Either a single element is refined (non-contextsensitive refinement) or a single element together with its environment is refined (contextsensitive refinement).

For a detailed description and further examples, the interested reader is referred to (Jaeschke, Oberweis, Stucky 1993).

3 Software Process Support

Software process support in the INCOME/STAR project has two major concerns. First is to provide developers with a guideline of how to perform information system development with INCOME/STAR. A framework called ProMISE (**Pro**cess **M**odel for **I**nformation **S**ystem **E**volution; cf. Scherrer, Oberweis, Stucky 1994) describes the methodology supported by INCOME/STAR and enables people involved in the development process to reflect, communicate and discuss the process. Its basic structure is lined out in Section 3.1.
Section 3.2 deals with the second concern of software process support in INCOME/STAR, which is Process model *enactment*. Active assistance is offered to developers for monitoring of development activities, document and workflow management (cf. Oberweis 1994), control of project responsibilities, capacity planning etc.

3.1 Basic Structure of ProMISE

System development with ProMISE takes an evolutionary approach, i.e. development and maintenance of a system are done as a sequence of sub-projects. ProMISE combines the advantages of a well-structured, stagewise approach to software development with other useful techniques, such as incremental refinement of documents, software reuse, prototyping, and cooperation support.
Figure 7 gives a graphical representation of a generic development stage in ProMISE. A specific design document ('result type') has a certain status as, e.g., requirements schema, implementation module, etc. and is modified with stage specific activities (e.g. semantic data model editing, compilation, etc.). In the graphical representation activity types have a specific gray level indicating the activity's degree of formality.

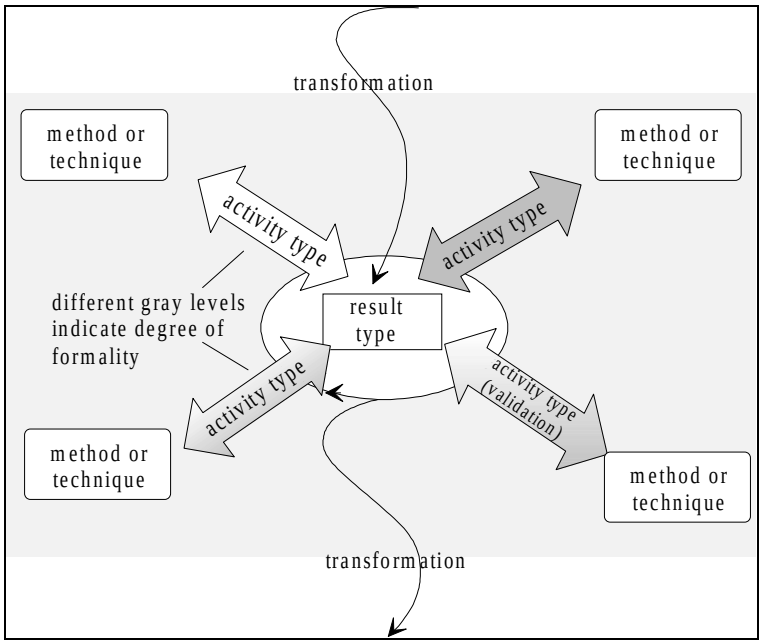

Fig. 7: Graphical representation of a generic development stage in ProMISE

Usually, document creation starts with a - more or less formal - transformation step, converting documents of the preceding stage into (initial) documents of the current phase. Next, documents are iteratively adjusted by an activity sequence (refinement, structuring, modeling, information collection steps etc.). If there are possible design alternatives, analytical methods or simulation may be used as a decision support. Software reuse is one potential alternative - either as an integration of standard components or as project specific adjustment of generic models.

At the end of each iteration, quality checks validate the results of transformation and modification steps. Whenever it makes sense, end users will be involved in this process. When a document's quality is acceptable, it may be transformed into an initial document of the succeeding stage. Otherwise a new iteration of information collection, modification and quality checking steps starts. Sometimes a situation may require a go-back to an earlier stage, e.g. if requirements are added or changed.
A specific description for *strategic planning, project specific planning, requirements collection and analysis, conceptual modeling, database design and implementation, program design and implementation* is available in (Scherrer, Oberweis, Stucky 1994).

As an example, Figure 8 shows requirements collection and analysis in the introduced notation.

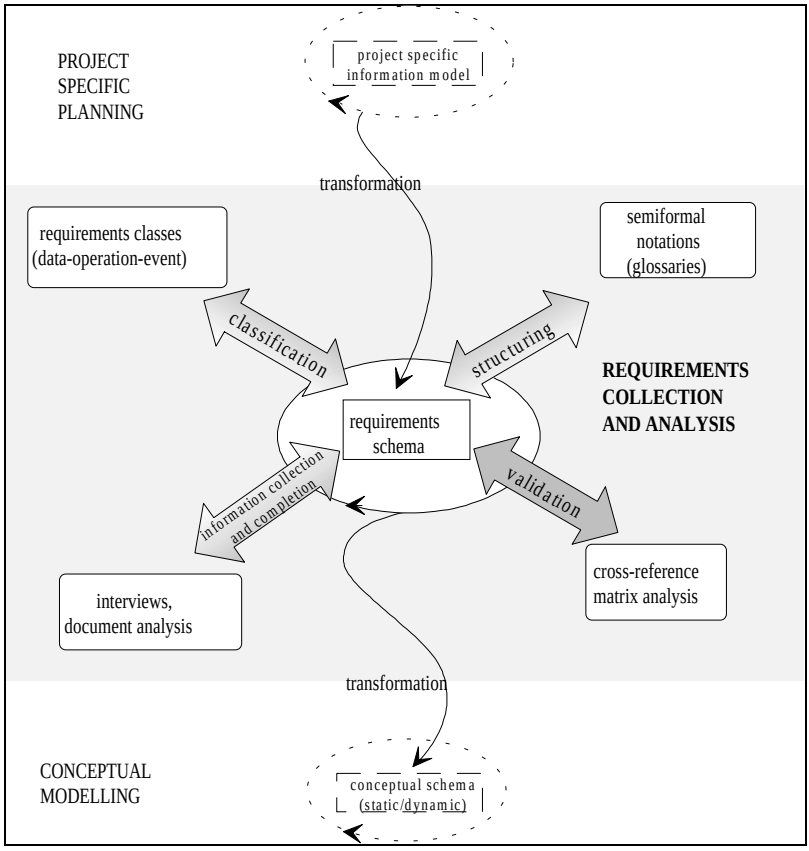

Fig. 8: Requirements collection and analysis stage

In the *initial transformation step*, a requirements collection plan is worked out by extracting business units and tasks from the *project specific information model*, which is the deliverable of the preceding stage. This initial document is refined to a complete *requirements schema* by iterating the following activity sequence:
For each task/business unit combination in the requirements collection plan, *information is collected and completed* through interviews and an analysis of existing documents. Next, information items are *classified* as data, operation or event and then *recorded in structured glossaries.* Cross-reference matrix analysis is used for *validation*. If any inconsistency or incompleteness is detected, a new iteration is performed.

3.2 Process model enactment

To obtain a computer supported, enactable version of ProMISE, it is specified as a hierarchy of Petri nets. A similar notation as in Figure 8 may be used on the top level. This top level representation provides a gross overview about the process model and may be used for communication between different groups of people involved in a software project. Manual and unstructured activities (like unstructured communication) are expressed by informal types of Petri nets inscribed with natural language expressions and enriched with icons that are easy to understand.
Stepwise refinement leads to a precise NR/T-net description of the process. The resulting nets are instantiated with a project-specific marking and can then be executed by a Petri net interpreter. Instantiation includes, e.g., association of activities and roles, roles and team members, deliverables and deadlines etc.
Process model enactment in INCOME/STAR means active process support. The enacted process model is coupled to the central repository through a *process engine* (a Petri net interpreter) which can control access to tools and data and manage the flow of information between people and tools involved in the software development process.

4 Advanced Simulation and Prototyping Concepts

In INCOME/STAR, simulation is an integrated part of the development process: simulators are interfaced with the central repository where the formal behavior specification is stored as a set of high-level Petri nets. Due

to the formal semantics of our underlying net model, this specification is directly executable.
Some innovative issues of the INCOME/STAR simulation and prototyping capabilities are now described in detail.

- **Simulation support for evolutionary development**

The simulation concepts in INCOME/STAR (cf. Mochel, Oberweis, Sänger 1993) support the evolutionary development approach prescribed by the chosen software development process model ProMISE: a preliminary system behavior specification - given as a set of high-level Petri nets - is simulated and analyzed by a novel graphical query language (see next paragraph). As a result of this validation step the Petri net model is improved. The same procedure is executed for the resulting net, probably in several cycles, until the system behavior is modeled adequately.

- **Graphical query language for large simulation runs**

Practical experience showed that for large Petri net models, simulation runs which are generated by automatic simulation 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.
Our novel graphical query languageGTL (**G**raphical **T**emporal **L**anguage) (cf. Oberweis, Sänger 1994) for simulation databases combines capabilities of temporal and graphical database languages. In the simulation database each net marking is interpreted as a single database state. GTL-queries are employed to check a simulation database for certain behavior patterns. These patterns may be related to single states (e.g. *Is there a simulation state where condition c1 holds?*) or to sequences of database states (e.g. *Is there a simulation state sequence where first c1 holds, then c2, and finally c3?*). In GTL simulation states are graphically represented by circles and temporal relationships between simulation states can be expressed along an implicit time axis. So-called checkpoints (graphically represented as boxes) specify complex conditions to select a state (sequence) in the simulation database.
For a detailed description of GTL see (Oberweis, Sänger 1994).

- **Multiuser environment and application specific visualizationof Petri net simulation**

An open simulation environment is provided which supports multiuser enactment and a coupling to external visualization devices.

When dealing with large systems, several developers are involved in the design process of the corresponding Petri net model. Therefore, the applied tools should include appropriate multiuser support:

- Access control for Petri net models to avoid inconsistencies when multiple developers try to apply changes to the net at the same time.
- A possibility to visualize the simulation run on an arbitrary number of workstations.
- Developers should be enabled to influence the simulation run decentralized from their own workstation.

Another useful property of a simulation environment is *application specific visualization.* A common drawback of most today's graphical Petri net simulation tools is that they provide an animated view of the transition occurrences only in the graphical representations of the Petri net itself, i.e. they visualize the flow of information along the arcs of a net. For large systems, a visualization on this level is not particularly useful because with increasing net complexity it becomes more and more difficult to imagine how a certain system state translates to 'reality'. So what is needed is an open simulation environment integrating arbitrary visualization modules which can provide a problem oriented display of the current system state.

A prototype called GAPS (**G**raphical **A**nimated **P**etri net **S**imulator) (cf. Oberweis, Sänger, Weitz 1994) implements these ideas: A person who initializes a simulation run becomes the 'master' of this process and can permit others to join in, either passively by letting them watch the simulation process or actively by granting them the right to influence the process. Beyond this external visualization, clients can register for certain events:

- The simulation starts.
- A given transition fires.
- The content of a given place changes.
- The simulation ends.

During simulation, the client is notified of the events it is registered for and reacts to such messages by updating its displays accordingly.

5 Cooperative System Design

Teamwork is an effective way to cope with the increasing complexity of software systems and high quality demands. Therefore, development environments for large software engineering projects should provide support for *cooperative development* work.

Teamwork in a software engineering project includes different coordination aspects. Process models like ProMISE imply different workflows as they support parallel processing of development deliverables. During the whole project, communication support and an efficient management of the flow of work items between different people or groups of people is required.
While process and workflow management primarily deal with technical aspects of software development, communication support concentrates on the *social action perspective.*

A framework of a role-based groupware system called RoCoMan (**Ro**le **Co**llaboration **Man**ager) supporting communication, organization, and social interaction was worked out. The following teamwork support components are currently coupled to the INCOME/STAR repository (cf. Oberweis, Wendel, Stucky 1994):

- **extended eMail system**

The eMail system maintains a semi-structured message exchange which supports message filtering methods to avoid information overload, a typical problem of existing computer-mediated communication systems. Therefore we have extended the eMail system by a component which interprets rule expressions like 'if mail arrives from team member Smith then put mail into dictionary smithMail'.
Furthermore there are four different types of eMail: common, formatted, extended and conversational eMail. Common eMail corresponds to conventional eMail. Formatted eMail supports a structuring of the mail content. Extended eMail allows the declaration of specific message types, for instance a request or a question. Conversational eMail is embedded in a so-called conversation which declares valid sequences of messages which can be modeled by a conversation editor.
Main components of the extended eMail system are a mail desktop and a mail editor. The desktop provides access to incoming mails and supports filtering methods for analyzing incoming mails. The editor maintains the creation of mail with respect to the selected eMail type. Additionally, it permits the transformation of a mail of a given type into another type.

- **day planner**

The day planner maintains three kinds of electronic calendar: personal, group and common project calendar. A personal electronic calendar consists of private and shared spaces. Shared spaces - in contrast to private spaces - are periods of time which can be read and manipulated by other authorized team members.

Shared spaces of team members are used to arrange group appointments. These appointments are registered in a group calendar. Furthermore, all deadlines and project dates are registered in a common project specific day plan which is accessible by all group members.

- **conversation manager**

The conversation managerallows planning and modeling of conversations and monitors progress information about current conversation processes.
Conversations are represented by *conversation diagrams*, a semi-formal graphical language which allows to specify communication processes in an easy and intuitive way (cf. Oberweis, Wendel, Stucky 1994). Each conversation consists of a *conversation act* representing a team member conversation activity, and a *processing relation* which links the conversation act to other conversation acts. A processing relation represents so-called conditions of completion to execute a conversation act: the importance (*conversational relevance*) of the conversation act, the *personal competence* and the *organizational role* of the team members performing the conversation act.

6 Related Work

Several other approaches propose software development environments supporting process model enactment and/or teamwork coordination. Some of them use different variants of Petri nets as a formal basis for the specification of system behavior:
MELMAC is a software process management environment using an extension of high-level Petri nets called FUNSOFT nets. MELMAC supports process analysis and process model enactment (cf. Deiters, Gruhn 1994).
SPADE-1 is a process-centered software engineering environment supporting software process analysis, design and enactment (cf. Bandinelli, Braga, Fuggetta et al. 1994). The underlying process modeling language - SLANG - is based on an extension of a high-level Petri net formalism called ER nets (cf. Ghezzi, Mandrioli, Morasca et al. 1991).
(Ellis, Nutt 1993) use Petri nets as a formal basis for the specification of workflows. Workflow management systems are similar to active process support systems in a sense that both types of systems support "information logistic" (cf. Fernström 1993), i.e. they support the flow of information between actors.
There are further proposals for process modeling languages combining semantic data modeling with Petri net based behavior modeling (cf. Eder,

Kappel, Tjoa et al. 1987; Heuser, Peres, Richter 1993; Lausen 1988; Sakai 1983; Sølvberg, Kung 1986).
However, none of these approaches provides appropriate concepts for behavior modeling of complex structured objects. There is no possibility to model concurrent access to different components of the same complex structured object. Our concept of NR/T-nets on the other hand supports locking with different granularity and by this allows to model concurrent access to design documents at different levels.

Several other approaches focus on social aspects of software projects:
(Hahn, Jarke, Rose 1990) 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.
Process WEAVER is a set of tools that adds process support to UNIX-based environments (cf. Fernström 1993). A notation similar to Petri nets is used to describe the control flow of process model activities. Process enactment assists in managing the flow of information in development teams, providing team members with task-specific work-contexts and automating certain activities.
(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 prototyping. However, it does not provide a graphical representation of workflows.
ConversationBuilder is a tool for collaborative software development described in (Kaplan, Tolone, Carroll et al. 1992). Based on ConversationBuilder (Gintell, Arnold, Houde 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.
In the ALF environment, a programming-language like construct called MASP (**M**odel for **A**ssisted **S**oftware **P**rocess) serves as process description formalism. A process model is described by a hierarchy of MASPs and can be therefore viewed at different levels of abstraction - a similar concept as the NR/T-net hierarchies in INCOME/STAR. There is, however, no facility for graphical visualization of MASPs. Efforts are made to apply ALF

technology to groupware support. As a first experiment, a conversation manager was built (cf. Lonchamp 1992).

7 Practical Experience and Outlook

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

Since some of the methods and tools are still under development, a practical evaluation of the complete system is not possible so far. Still, some valuable experiences were gained by evaluating parts of the system in some smaller case studies:
An information system for the administration of examination data was entirely developed with methods and tools available in the INCOME/STAR development environment (cf. Jaeschke, Stucky 1994).

Several other case studies were carried out in cooperation with external partners to determine practical requirements and gain experiences with some specific methods. Two questions were of particular interest:

① Which semantic data model is most applicable in practice and should therefore be supported by the INCOME/STAR methodology? What is more important in practice: rich semantics or simplicity? Three alternatives were taken into consideration:

- a simple binary Entity-Relationship model
- an extended Entity-Relationship model
- the semantic hierarchy object model

One (surprising) result was that it can make sense to use different variants of the ER model in the same project. In spite of its restricted expressiveness, the simple, binary variant seems to be an adequate basis for discussion with end users, while versions with enriched semantics are preferred by software experts. But even developers sometimes switch to the binary variant at later stages, mainly because it can easily be converted into a relational database schema.
The semantic hierarchy object model seems to fit best with the NR/T-net concept and NF^2 databases.
For this reason, instead of restricting the INCOME/STAR environment to one data model, we are thinking about a component which supports a conversion from one model to another.

② Is there a reasonable degree of acceptance for Petri nets in practice? How should methodological support for Petri nets look like?
Experiences in this area were quite contradictory: Acceptance for Petri nets seem to be much better in manufacturing than in administration. One possible explanation for this phenomenon could be that behavioral aspects of technical processes are more obvious and can therefore be modeled more easily. There is a lack of methodological support for the development of Petri net models, especially for applications in administration, where behavioral aspects can normally not be recognized as intuitively as in technical applications. Most advantageous for practical acceptance seem to be user-friendly visualization techniques and an automated generation of Petri nets and markings.

Future research work includes the following issues: While our graphical editors support both Pr/T-nets and NR/T-nets, our simulators currently work with strict Pr/T-nets. For the future, both net types will be supported. Furthermore, we are planning methodological support for a conversion from one net type into the other.
Our support for process modeling currently concentrates on qualitative aspects of software process management. Now we are planning to consider quantitative aspects as well by adding a component for software productivity and quality measurement.
As far as teamwork coordination is concerned, an important aspect of future research is the support of other system environments, e.g. available commercial groupware applications.
A final research direction concerns the replication of parts of the repository in distributed environments, which is not yet supported.

References

Bandinelli, S.; Braga, M.; Fuggetta, A.; Lavazza, L.: The architecture of the SPADE-1 process-centered SEE; in: Warboys, B.C. (Ed.): Software Process Technology, Springer 1994, pp.15-30.

Barghouti, N.S.: Supporting cooperation in the MARVEL process-centered SDE; ACM SIGSOFT Software Engineering Notes, 17(5), 1992, pp. 21-31

Barker, R.: CASE*Method: Tasks and Deliverables; Addison-Wesley 1990.

Berztiss, A.T.: Concurrent engineering of information systems; in: Prakash, N.; Rolland, C.; Pernici, B. (Eds.): Information System Development Process, North-Holland 1993, pp. 311-324.

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

Deiters, W.; Gruhn,V.: The FUNSOFT net approach to software process management; International Journal on Software Engineering and Knowledge Engineering, 4(2), pp. 229-256 1994.

Eder, J.; Kappel, G.; Tjoa, A.M.; Wagner, A.A.: BIER - The behaviour integrated entity relationship approach; in: Spaccapietra, S. (Ed.): Proc. 5th International Conference on the Entity-Relationship Approach, North-Holland 1987, pp. 147-168.

Ellis, C.A.; Nutt, G.J.: Modeling and enactment of workflow systems; in: Marsan, M.A. (Ed.): Proc. 14th International Conference on Application and Theory of Petri Nets, Springer 1993, pp. 1-16.

Feldman, P.; Miller, D.: Entity model clustering: Structuring a data model by abstraction; The Computer Journal, 29(4), 1986, pp. 348-360.

Fernström, C.: PROCESS WEAVER: Adding process support to UNIX; Proc. 2nd International Conference on the Software Process, IEEE Computer Society Press 1993, pp. 12-26.

Genrich, H.J.; Lautenbach, K.: System modelling with high-level Petri nets; Theoretical Computer Science, 13, 1981, pp. 109-136.

Ghezzi, C.; Mandrioli, D.; Morasca, S.; Pezzè, M.: A unified high-level Petri net formalism for time-critical systems; IEEE Transactions on Software Engineering, 17(2), 1991, pp. 160-172.

Gintell, J.; Arnold, J.; Houde, M.; Kruszelnicki, J.; McKenney, R.; Memmi, G.: Scrutiny: A collaborative inspection and review system; in: Sommerville, I.; Paul, M. (Eds.): Proc. Software Engineering - ESEC'93, Springer 1993, pp. 344-360.

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

Heuser, C.A.; Peres, E.M.; Richter, G.: Towards a complete conceptual model: Petri nets and entity-relationship diagrams; Information Systems, 18(5), 1993, pp. 275-289.

INCOME User Manuals: INCOME/Designer, INCOME/Dictionary, INCOME/Generator, INCOME/Simulator; PROMATIS Informatik, Karlsbad/Germany 1994.

Jaeschke, P.; Oberweis, A.; Stucky, W.: Extending ER model clustering by relationship clustering; in: Elmasri, R.; Kouramajian, V. (Eds.): Proc. 12th International Conference on the Entity-Relationship Approach, Arlington/Texas 1993, pp. 447-459.

Jaeschke, P.; Stucky, W.: An integrated tool for information system development: practical experience; Universität Karlsruhe, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Forschungsbericht 297, Karlsruhe/Germany 1994.

Kaplan, S.M.; Tolone, W.J.; Carroll, A.M.; Bogia, D.P.; Bignoli, C.: Supporting collaborative software development with Conversation-Builder; ACM SIGSOFT Software Engineering Notes, 17(5), 1992, pp. 11-20

Lausen, G.: Modelling and analysis of the behaviour of information systems; IEEE Transactions on Software Engineering, 14(11), 1988, pp. 1610-1620.

Lausen, G.; Németh, T.; Oberweis, A.; Schönthaler, F.; Stucky, W.: The INCOME approach for conceptual modelling and prototyping of information systems; Proc. 1st Nordic Conference on Advanced Systems Engineering, Stockholm/Sweden 1989.

Lonchamp, L.: Supporting social interaction activities of software processes. in: Derniame, J.C (Ed.): Software Process Technology, Springer 1992, pp. 34-54.

Mistelbauer, H.: Datenmodellverdichtung: Vom Projektdatenmodell zur Unternehmensarchitektur; Wirtschaftsinformatik, 33(4), 1991, pp. 289-299.

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

Oberweis, A.: Workflow management in software engineering projects; in: Medhat, S. (Ed.): Proc. 2nd International Conference on Concurrent Engineering and Electronic Design Automation, Bournemouth/UK 1994, pp. 55-60.

Oberweis, A.; Sänger, V.: Graphical query language for simulation runs; Journal of Microcomputer Applications, 17, 1994.

Oberweis, A.; Sänger, V.; Weitz, W.: GAPS - A multiuser tool for graphical simulation of Petri Nets; in: Halin, J.; Karplus, W.; Rimane, R. (Eds.): Proc. 1st Joint Conference of International Simulation Societies, Zurich/Switzerland 1994, pp. 377-381.

Oberweis, A.; Sander, P.; Stucky, W.: Petri net based modelling of procedures in complex object database applications; in: Cooke, D. (Ed.): Proc. IEEE 17th Annual International Computer Software and Applications Conference, Phoenix/Arizona 1993, pp. 138-144.

Oberweis, A.; Wendel, T.; Stucky, W.: Teamwork coordination in a distributed software development environment; in: Wolfinger, B. (Ed.): Innovationen bei Rechen- und Kommunikationssystemen, Springer 1994, pp. 423-429.

Rauh, O.; Stickel, E.: Entity tree clustering - a method for simplifying ER design; in: Pernul, G.; Tjoa, A.M. (Eds.): Proc. 11th International Conference on the Entity-Relationship Approach, Springer 1992, pp. 62-78.

Reisig, W.: Petri Nets; EATCS Monographs on Theoretical Computer Science, Springer 1985.

Sakai, H.: A method for entity-relationship behaviour modeling; in: Davis, C.G.; Jajodia, S.; Ng, P.A.; Yeh, R.T. (Eds.): Entity-Relationship Approach to Software Engineering, North-Holland 1983, pp. 111-129.

Schek, H.-J.; Scholl, M.: The relational model with relation-valued attributes; Information Systems, 11(2), 1986, pp. 137-147.

Scherrer, G.; Oberweis, A.; Stucky, W.: ProMISE - a process model for information system evolution; Proc. 3rd Maghrebian Conference on Software Engineering and Artificial Intelligence, Rabat/Morocco 1994, pp. 27-36.

Sølvberg, A.; Kung, D.C.: On Structural and behavioral modelling of reality; in: T.B. Steel and R. Meersman (Eds.): Database Semantics, North-Holland 1986, pp. 205-221.

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

—C—

—D—

—E—

—F—

—G—

—I—

—N—

—O—

—P—

—R—