INCOME/STAR: Methodology and Tools for the Development of Distributed Information Systems

Andreas Oberweis, Gabriele Scherrer, Wolffried Stucky[1]
Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe (TH)
D-76128 Karlsruhe
Germany

eMail: {oberweis|scherrer|stucky}@aifb.uni-karlsruhe.de

Abstract

INCOME/STAR is an experimental environment for the cooperative development of distributed information systems. This paper presents some of INCOME/STAR's innovative features in the area of information systems engineering:

First a new type of high-level Petri nets, so-called Nested Relation/Transition nets (NR/T-nets), is described. NR/T-nets allow the modeling of concurrent processes and related complex structured objects in distributed business applications.

New concepts for *entity* and *relationship clustering* were developed to support a stepwise top-down approach for Entity/Relationship based object modeling. Distributed *multi-user simulation and prototyping* are used for the evaluation and analysis of NR/T-nets and the involved object schema.

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

1 Introduction

The efficient production of high-quality information systems has been - and still is - a major objective of information systems engineering [47]. But an industrialization of software engineering in general and information systems engineering in particular is still 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 the following aspects:

[1] Our colleagues Peter Jaeschke, Volker Sänger, Wolfgang Weitz, and Thomas Wendel have substantially contributed to the work described in this paper.

Complexity: Information systems are not only supposed to be particularly 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 usually integrated in networks.

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

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

The increasing complexity of the target systems causes high demands to information system development environments. Recent efforts in this area have largely been concentrating on two aspects:

① *Control for the software process:* A great deal of research has been done lately to achieve a deeper understanding of the software development process [11, 26, 39, 49] which can be defined as a set of activities, methods and principles that guide people in the production of software [28]. Several so-called process-centered development environments have been proposed, e.g. [1, 2, 4, 8, 10, 15]. The idea is to model the development process in an executable notation in order to automate parts of it and provide guidance and support to developers during process execution [1].

② *Cooperative design techniques:* A major part of software development work is done in teams. Therefore, some development environments offer groupware functionality and support of social interaction [2, 5, 15, 20, 27, 31]. [19] discusses aspects of concurrency control related to cooperative system development from a more technical viewpoint.

Although such efforts - software process control and teamwork support - can contribute substantially to an efficient support of information system development, they do not solve all problems that arise from the development of really complex systems.
First, *methodological aspects* have to be considered as well: A development environment must
support advanced methods which reflect the distribution and complexity of objects and processes of the target system, i.e. techniques for an integrated modeling of distributed, cooperative processes and related complex structured objects are needed.

A second aspect is assistance for *evolutionary system development* and *development support for heterogeneous systems.* For this purpose, simulation can be a valuable help. It is particularly useful in a multi-user environment with application specific visualization techniques.
The objective of the INCOME/STAR project[2] was to conceive an information system development environment that combines process and teamwork support with methods for integrated modeling of system behavior and related complex structured objects and advanced simulation techniques. A prototype as been implemented which is briefly described in the next section.
The remainder of the paper summarizes the most important new concepts of INCOME/STAR, grouped into four research directions: new methodological concepts (Section 3), software development process support (Section 4), advanced simulation and prototyping concepts (Section 5) and cooperative system design (Section 6). Related approaches are discussed in Section 7 while the final section surveys first practical experiences and gives a short outlook on future research work.

2 The INCOME/STAR Environment

The INCOME/STAR prototype is based on INCOME (**I**nteractive **N**et-based **Co**nceptual **M**odeling **E**nvironment), a 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 university version of INCOME are: integration of structural and behavioral system aspects, prototyping facilities and design dictionary support [30].
Based on these concepts, a commercially available methods and tools package was developed. The commercial product INCOME [23] is embedded in the ORACLE*CASE product family. ORACLE*CASE supports CASE*Method, an information engineering approach for database oriented system development [3]. 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.

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).
The main components of INCOME/STAR are:

[2] The project is partially supported by the Deutsche Forschungsgemeinschaft DFG under grant Stu 98/9 in the program "Verteilte DV-Systeme in der Betriebswirtschaft“.

- graphical editors for a *semantic object model*, for *high level Petri nets* and for other design documents (such as, e.g., *function hierarchies* and *object glossaries*).
- database and application program generators: the database generator produces a relational database definition (including forms and reports definitions) from the conceptual data schema. The application program generator produces C-Programs with embedded SQL-commands from the conceptual behavior schema.
- simulation and prototyping facilities based on high-level Petri nets (cf. Section 5).
- a repository to store development documents and to maintain consistency of documents.
- facilities for teamwork support (cf. Section 6).
- a component for development process modeling and enactment (cf. Section 4).

3 New Methodological Concepts

Modeling highly flexible information systems requires object structures that are not as restricted as postulated by the relational data model. Therefore, the new methodological concepts of INCOME/STAR aim at providing 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 (**N**ested **R**elation/**T**ransition nets) - a new variant of high-level Petri nets closely related to NF^2 (**N**on **F**irst **N**ormal **F**orm) relational databases [43] - for behavior modeling (Section 3.1).
Another useful method 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 3.2).

3.1 Nested Relation/Transition Nets

In INCOME/STAR Petri nets[3] are used as the formalism for behavior modeling because of the following characteristics:

- on the one hand, the well-defined formal semantics for Petri nets allows the application of different analysis methods such as discovering deadlocks or 'dead code', i.e. operations that will never be executed, etc.
- on the other hand, Petri nets are an executable specification of the target system. By using the Petri net simulator presented below, the developer obtains an early impression of the system behavior. The simulation of the Petri net model can also be used as a basis for discussion with the enduser community.

[3] We suppose that the reader is familiar with the basic Petri net notation (see, e.g., [41]).

INCOME/STAR supports a new type of high-level Petri nets, namely nested relation/transition nets (NR/T-nets) [37]. To each place in an NR/T-net, a complex structured object type is assigned, specified in a semantic data model similar to SHM (**S**emantic **H**ierarchy **M**odel) [6]. Basic constructs for data structuring are classification, aggregation, specialization and grouping. Figure 3.1-1 shows the graphical representation of these concepts.

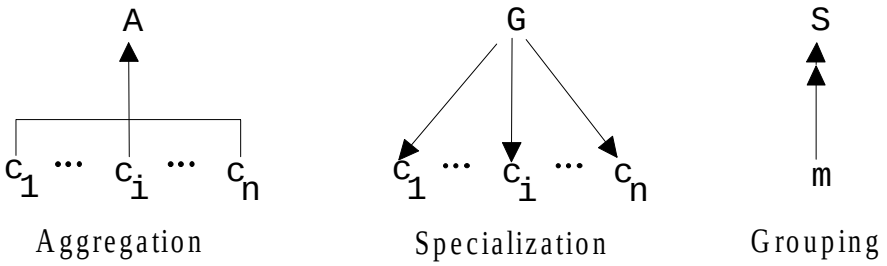

Figure 3.1-1: Structuring concepts in SHM [6]

The marking of a place in an NR/T-net is a nested relation of the respective type, i.e. a set of complex structured 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)* [16]: 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 structured object. An example is a situation where different project team members access different parts of the same document.

Example

Figure 3.1-2 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.

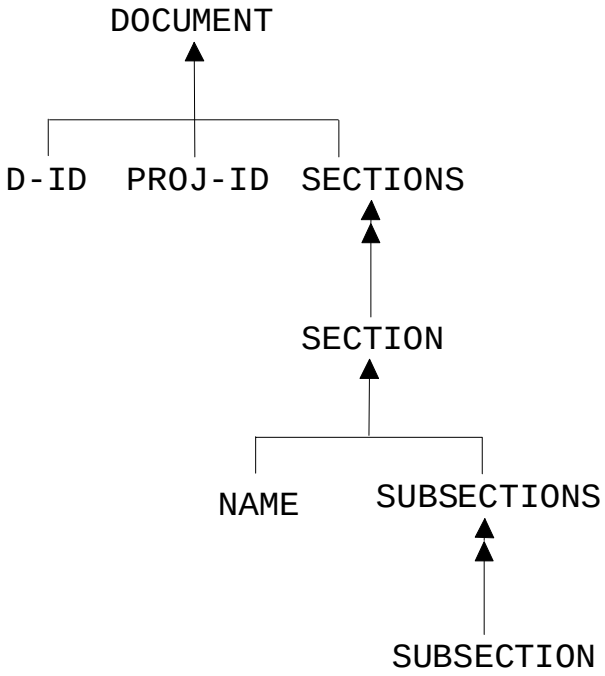

Figure 3.1-2: Type DOCUMENT

Figure 3.1-3 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}> }

Figure 3.1-3: Tabular representation of three example objects of type DOCUMENT

Figure 3.1-4 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 3.1-3.

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.

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 into 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 3.1-4, always access complete attribute values. In Figure 3.1-4, $\overline{X}$ 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}>     }.
```

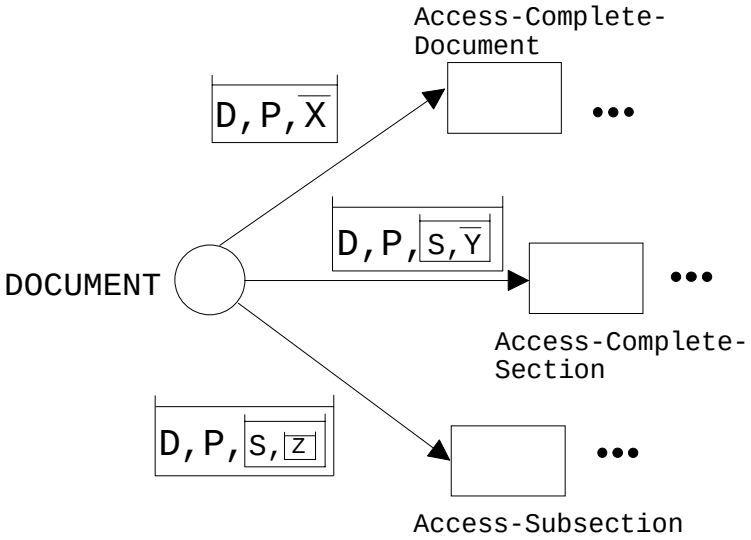

Figure 3.1-4: Example NR/T-net (A)

So-called *open* variables, e.g. X in Figure 3.1-5, 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}> }.
```

Filter tables may be hierarchically structured to reflect the hierarchic structure of complex objects. This allows access to values which are located on the lower levels of the attribute hierarchy.

In Figure 3.1-4 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 given in Figure 3.1-5 is as follows:

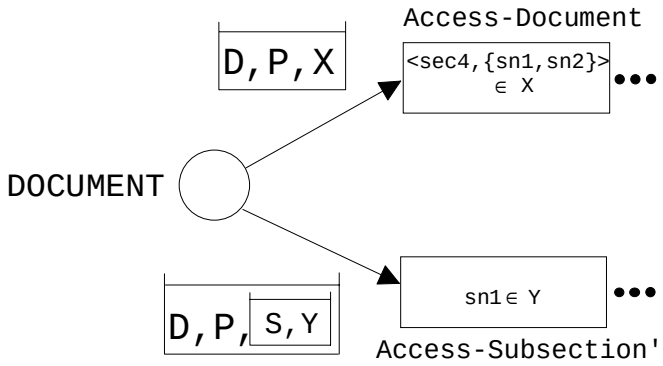

Figure 3.1-5: Example NR/T-net (B)

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

For a detailed description and further examples, the interested reader is referred to [37].

3.2 ER Model Clustering

The Entity-Relationship approach [7] 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, e.g., neither a way to obtain a general view nor 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 [14, 32, 40, 48]. Whole 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 [24]:

① **Entity clustering** was first proposed in [14]. 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 detailed original entity types. The higher level diagram is iteratively abstracted 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 3.2 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.

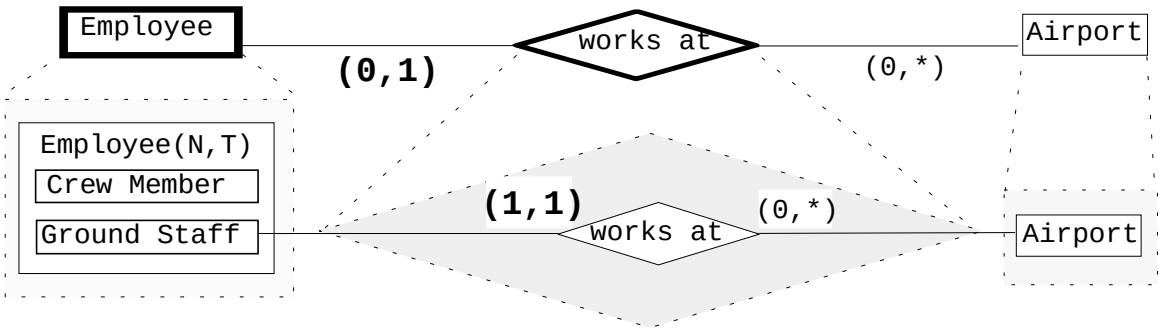

Figure 3.2: Refinement of 'Employee' based on entity clustering; refinement of 'works at' based on simple relationship clustering

Simple relationship clustering can also be applied to represent integrity constraints in an ER diagram and to cluster semantically similar relationship types into one.

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, but 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 [24].

4 Software Process Support

Within the context of the INCOME/STAR project, software development process support has two major concerns. The first one 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) [44] describes the methodology supported by INCOME/STAR as well as organizational and cooperative aspects of the development process. Its basic structure is lined out in Section 4.1.

A process model *description* is a useful concept as it enables people involved in the development process to reflect, communicate and discuss the process. The second concern of software process support in INCOME/STAR goes even further: Process model *enactment* can offer active assistance to developers for monitoring of development activities, document and workflow management [34], control of project responsibilities, capacity planning etc. (cf. Section 4.2).

4.1 Basic structure of the development process

System development with ProMISE takes an evolutionary approach, i.e. development and maintenance of a system are performed 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 4.1-1 surveys the basic structure of ProMISE. A graphical representation of a generic development stage is shown. 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.

Activities concerning different parts of the system may be carried out simultaneously by different persons, i.e. stages may be processed in parallel or overlap. Such cooperative aspects are reflected in more detailed representations of the process model.

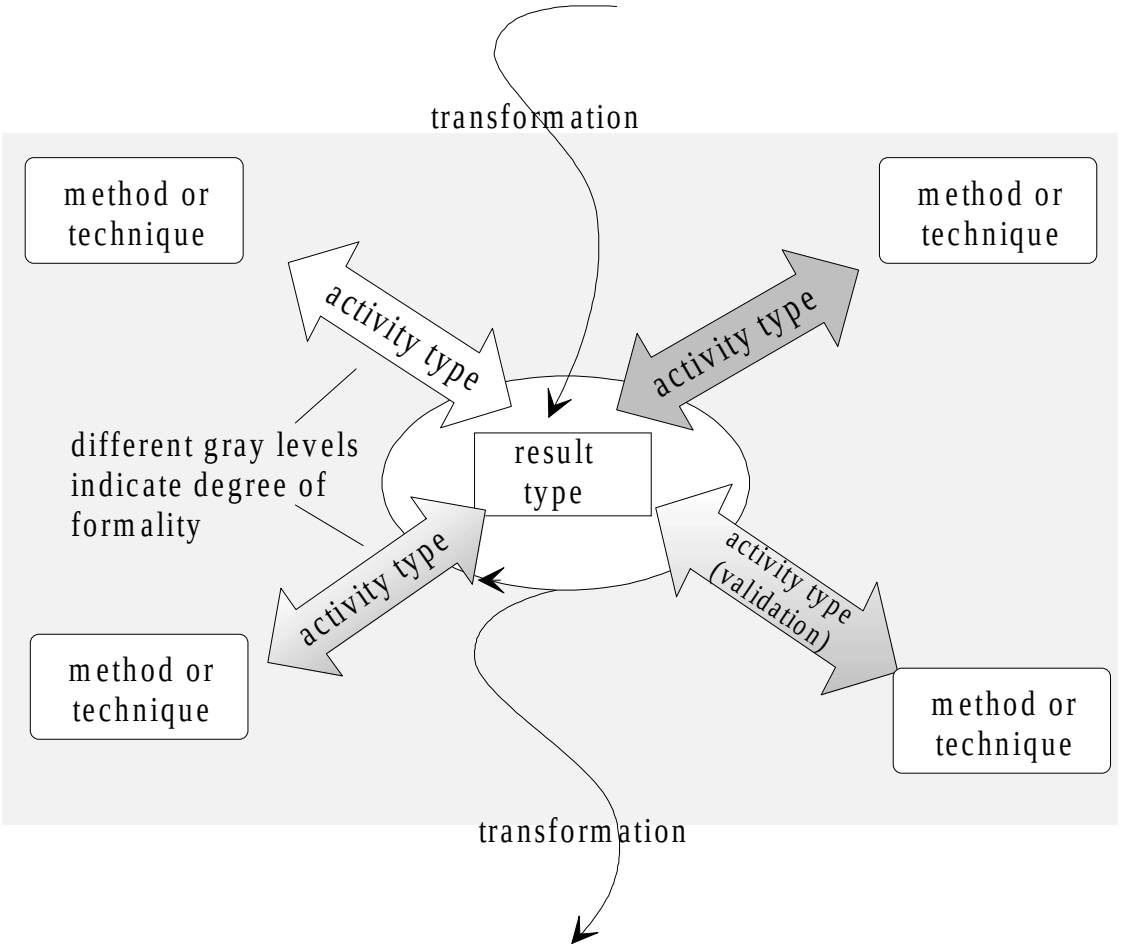

Figure 4.1-1: Graphical representation of a generic development stage in ProMISE

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 modified by a sequence of activities (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 a project specific adjustment of generic models.

At the end of each iteration, quality checks validate the results of transformation and adjustment 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 rollback to an earlier stage for a certain document, 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 [44].

As an example, Figure 4.1-2 shows requirements collection and analysis in this notation.

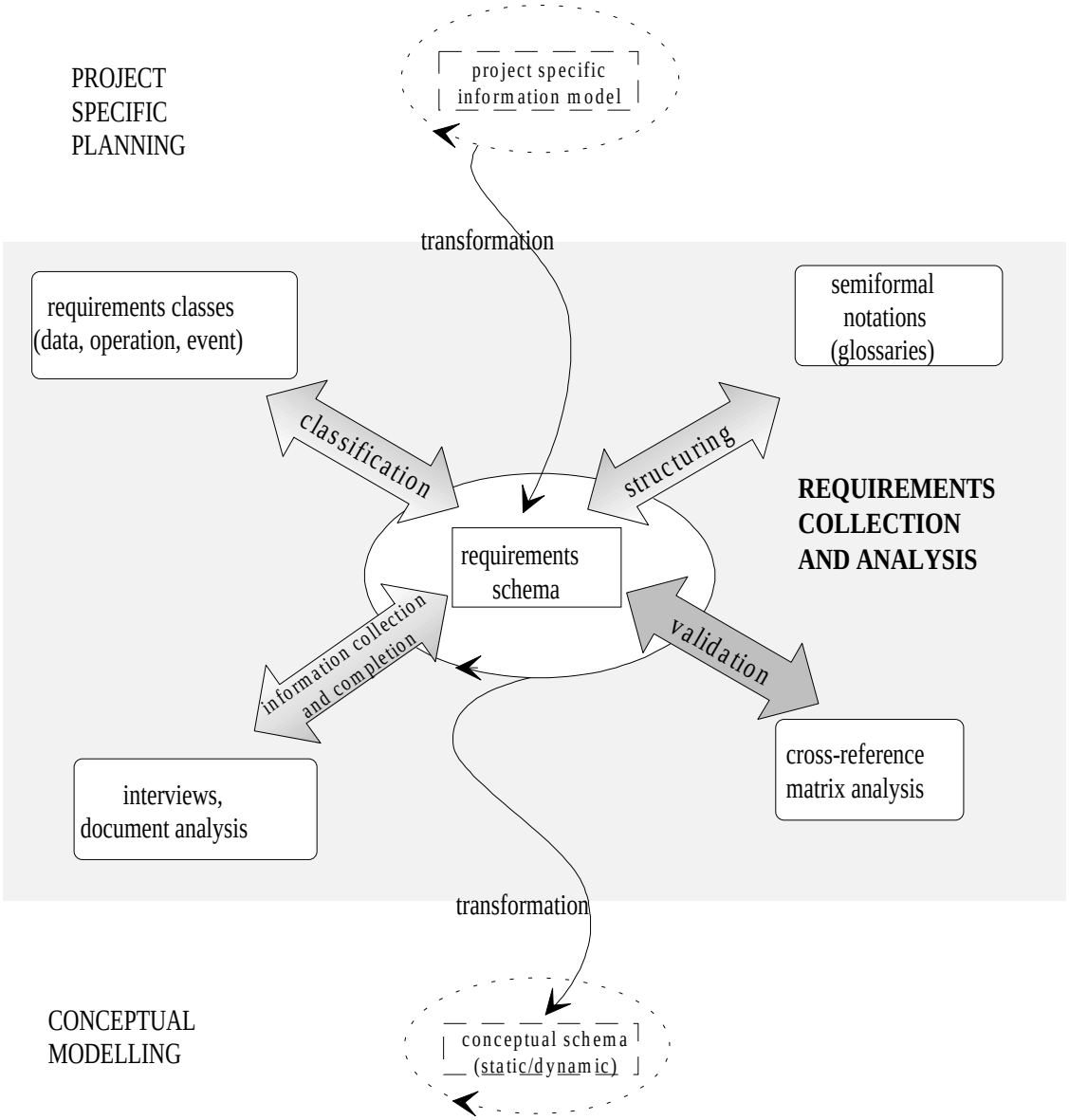

Figure 4.1-2: Requirements collection and analysis stage

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 (cf. [9]). Cross-reference matrices provide automated quality-checking facilities validating integrity rules like '*if there exists an object x in document d1, document d2 must contain an operation y*'. If any inconsistency or incompleteness is detected, a new iteration is performed.

4.2 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 4.1-2 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.

5 Advanced Simulation and Prototyping Concepts

In INCOME/STAR, simulation is an integral part of the development process: simulators are interfaced with 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 our underlying net model, this specification is directly executable.

The INCOME/STAR simulation and prototyping capabilities are now described in detail.

- **Simulation support for evolutionary development**

The simulation concepts in INCOME/STAR [33] 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). Due to this validation step improvements and corrections are integrated into the Petri net model. The same procedure is executed for the resulting net, probably in several cycles, until the system behavior is modeled adequately.

One possible strategy for evolutionary development is to distinguish between different potential system behaviors. It is useful to concentrate first on those system procedures which are regarded as being important, i.e. to abstract from system procedures which are regarded to be less important in the early design stages. The distinction between important and less important system procedures must be done by the system designer and is a rather subjective

decision. Important system procedures may be e.g. regular system procedures, less important system procedures may be exceptional procedures which occur e.g. in (usually) rare error cases.

For system behavior which is specified in a Petri net this strategy means: In early design stages it is distinguished between regular and irregular markings (system states) and between regular and irregular net marking sequences (system state sequences). First, only regular net behavior is considered. In the following design stages, exceptional behavior is gradually integrated. The process of integration is supported by net simulation. To define irregular markings/marking sequences the concept of so-called *checkpoint transition/checkpoint transition sequence* is used.

A checkpoint transition is a special kind of transition whose enabledness for a certain marking indicates that this marking represents an exceptional situation. Different checkpoint transitions may be combined to so-called checkpoint transition sequences which denote exceptional marking sequences. An enabled checkpoint transition cannot change the current marking. Nevertheless, checkpoint transitions are useful at the design level for the specification of exceptional markings.

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

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. The designer of a system is interested in questions like the following: Does the simulation run contain

- any marking which is not allowed?
- every marking which must be reached?
- any marking sequence which is not allowed?
- every marking sequence which must occur?

In [35] the graphical query language for simulation databases GTL (**G**raphical **T**emporal **L**anguage) is proposed. GTL 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. Temporal relationships between simulation states can be expressed along an implicit time axis. Again, checkpoint transitions (checkpoint transition sequences) are used to specify complex conditions to select states (state sequences) in the simulation database.

For a detailed description of GTL see [35].

- **Development support for embedded heterogeneous systems**

To support the development of heterogeneous systems - consisting of components that may differ from each other with respect to user interfaces, communication protocols, operating systems, data formats and types of operation - it is sometimes useful to couple different simulators to each other or to couple simulators to external hardware or software devices.
Simulation support for the design process of heterogeneous embedded systems consists of three steps:

- The components are independently simulated and validated isolated from each other.
- Different simulation processes, each one representing one single component, are coupled to each other.
- Already implemented components (e.g. a database system or an application program) are coupled to the simulation processes.

Simulators can be coupled in different ways:

Loose Coupling:
Different simulators/components may be loosely coupled via storage devices (in main memory or secondary storage). A given simulator/component may, e.g., write data into a certain file which is later read by another simulator/component. Several mechanisms are supported for main memory communication by operating systems, such as pipes, sockets or mailboxes [45]. If more than two simulators are loosely coupled to each other, a blackboard mechanism is needed, where several simulators may write to and where several simulators may read from.

Procedure Calls:
Another type of coupling pairs of simulators is provided by the possibility to externally trigger certain procedures, which are to be executed by a simulator. Possibly the trigger is parametrized, i.e. it may carry certain additional information.

Central Control Unit:
The third type of coupling supports the coupling of more than two simulators/components. A central control unit manages all kind of communication between the connected simulators/components.

Several decisions are related to the coupling of different simulators: One decision is between user control and automatic control of simulation processes. This concerns the question of autonomy of simulators which are coupled to each other. Different levels of autonomy are possible from complete autonomy to complete dependency. The decision also concerns the management of simulation time. Finally, the different data formats must be transformed, such that different simulators and components understand each other. Two general architectures exist for data format transformations:

- Between each pair of interfaces a specific bilateral interface (*gateway*) is provided.

- A multilateral transformation component is introduced. Each message is first sent to this transformation component and then translated into a certain meta-language. The meta-language expression is then translated into the target language.

The selection of one specific data transformation architecture obviously depends on the number of communication links which must be supported.

- **Multiuser environment and application specific visualization of 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 multi-user 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) [36] 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 allowing them to 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.

6 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 developed. The following teamwork support components are currently included in the INCOME/STAR environment [38]:

- **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.
The 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 manager allows 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 [38]. 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.

- **workflow manager**

The workflow manager supports planning and modeling of development activities based on the development process model. It monitors and controls the execution of workflows and supports resource allocation.
The workflow modeling component of INCOME/STAR consists of three NR/T-net based tools: a *workflow editor* to edit workflows and the corresponding object structures, a *workflow simulator* to visualize and validate workflow models and a *workflow analyzer* for consistency checking [34].

7 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, namely FUNSOFT nets. MELMAC supports process analysis and process model enactment [10].
SPADE-1 is a process-centered software engineering environment supporting software process analysis, design and enactment [1]. The underlying process modeling language - SLANG - is based on an extension of a high-level Petri net formalism called ER nets [17].
[13] 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" [15], 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 [12, 21, 22, 29, 42, 46].
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:
[20] 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 [15]. 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.
[5] 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 [27]. Based on ConversationBuilder [18] propose a collaborative inspection and review system for software engineering products. The system is tailorable to different development process models.
[2] 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 [31].
Another group of software development environments use object-oriented languages for process modeling, e.g. the SPELL language in the EPOS approach [8] or the TEMPO language in the Adele system [4].

8 First Practical Experiences and Outlook

INCOME/STAR is currently being implemented in a workstation environment. The user interface (including the graphical editors) is already realized in SMALLTALK, the simulation kernel in PROLOG. 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. However, some valuable experiences were gained by evaluating parts of the system in some smaller case studies:
An information system for the administration of examinations at our institute was developed [25]. This was entirely done with methods and tools available in the INCOME/STAR development environment. Although this project came too early to validate all concepts described in this paper, at least the conclusion could be drawn that there is a need for advanced methods like entity and relationship clustering or NR/T-net modeling. It also showed the usefulness of simulation within a development process.
Several other case studies were carried out in cooperation with industry 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 which is supported by the ORACLE*CASE environment (and therefore by the commercial version of INCOME),
- an extended Entity-Relationship model (cf. Section 3.2),
- the semantic hierarchy object model which was the data model originally used in the university version of INCOME resp. INCOME/STAR.

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 be easily 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 the computer integrated manufacturing area than in the business administration area. 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 business 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, the simulators currently only work with 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 to 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 functionality 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 development environments, which is not yet supported.

Acknowledgement
We wish to thank our colleagues Hans Richter, Volker Sänger, and Wolfgang Weitz for valuable discussions and many useful comments on this paper.

References

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

[2] N.S. Barghouti. Supporting cooperation in the MARVEL process-centered SDE. *Proc. 5th ACM SIGSOFT Symposium on Software Development Environments, Software Engineering Notes*, **17** (5), 21-31 (1992).

[3] R. Barker. *CASE*Method: Tasks and Deliverables.* Addison-Wesley (1990).

[4] N. Belkhatir, J. Estublier, and W.L. Melo. Software process model and work space control in the Adele system. *Proc. 2nd International Conf. on the Software Process*, Berlin, Germany, 2-11 (1993).

[5] A.T. Berztiss. Concurrent engineering of information systems. in *Information System Development Process*, N. Prakash, C. Rolland, and B. Pernici (Eds.), North-Holland, 311-324 (1993).

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

[7] P.P.S. Chen. The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, **1** (1), 9-36 (1976).

[8] R. Conradi, M.L. Jaccheri, C. Mazzi, M.N. Nguyen, and A. Aarsten. Design, use and implementation of SPELL, a language for software process modeling and evolution. in *Software Process Technology*, J.C. Derniame (Ed.), Springer, 167-177 (1992).

[9] V. De Antonellis and B. Demo. Requirements collection and analysis. in *Methodology and Tools of Database Design*, S. Ceri (Ed.), North-Holland, 9-24 (1983).

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

[11] J.C. Derniame (Ed.). *Software Process Technology*. Springer (1992).

[12] J. Eder, G. Kappel, A.M. Tjoa, and A.A. Wagner. BIER - The behaviour integrated entity relationship approach. *Proc. 5th International Conf. on the Entity-Relationship Approach*, S. Spaccapietra (Ed.), North-Holland, 147-168 (1987).

[13] C.A. Ellis and G.J. Nutt. Modeling and enactment of workflow systems. *Proc. 14th International Conf. on Application and Theory of Petri Nets*, Marsan, M.A. (Ed.), Springer, 1-16 (1993).

[14] P. Feldman and D. Miller. Entity model clustering: Structuring a data model by abstraction. *The Computer Journal*, **29** (4), 348-360 (1986).

[15] C. Fernström. PROCESS WEAVER: Adding process support to UNIX. *Proc. 2nd International Conf. on the Software Process*, IEEE Computer Society Press, 12-26 (1993).

[16] H.J. Genrich and K. Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, **13**, 109-136 (1981).

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

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

[19] C. Godard. COO: A transaction model to support COOperating software developers COOrdination. *Proc. Software Engineering - ESEC'93*, I. Sommerville and M. Paul (Eds.), Springer, 361-379 (1993).

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

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

[22] A. Horndasch, R. Studer, and R. Yasdi. An approach to (office) information system design based on general net theory. *Proc. IFIP TC8.1 TFAIS85*, North-Holland (1985).

[23] INCOME User Manuals. *INCOME/Designer, INCOME/Dictionary, INCOME/Generator, INCOME/Simulator*. PROMATIS Informatik, Karlsbad, Germany (1994).

[24] P. Jaeschke, A. Oberweis, and W. Stucky. Extending ER model clustering by relationship clustering. *Proc. 12th International Conf. on the Entity-Relationship Approach*, Arlington, Texas, 447-459 (1993).

[25] P. Jaeschke and W. Stucky. An integrated tool for information system development: practical experience. Forschungsbericht 297, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe (1994).

[26] M. Jarke and P. Pohl. Vision driven system engineering. in *Information System Development Process*, N. Prakash, C. Rolland, and B. Pernici (Eds.), North-Holland, 3-20 (1993).

[27] S.M. Kaplan, W.J. Tolone, A.M. Carroll, D.P. Bogia, and C. Bignoli. Supporting collaborative software development with ConversationBuilder. *Proc. 5th ACM SIGSOFT Symposium on Software Development Environments, Software Engineering Notes*, **17** (5), 11-20 (1992).

[28] D.H. Kitson and S.M. Masters. An analysis of SEI software process assessment results: 1987-1991. *Proc. 15th International Conf. on Software Engineering*, Baltimore, Maryland, 68-77 (1993).

[29] G. Lausen. Modelling and analysis of the behaviour of information systems. *IEEE Transactions on Software Engineering*, **14** (11), 1610-1620 (1988).

[30] G. Lausen, T. Németh, A. Oberweis, F. Schönthaler, and W. Stucky. The INCOME approach for conceptual modelling and prototyping of information systems. *Proc. 1st Nordic Conf. on Advanced Systems Engineering*, Stockholm, Sweden (1989).

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

[32] H. Mistelbauer. Datenmodellverdichtung: Vom Projektdatenmodell zur Unternehmensarchitektur. *Wirtschaftsinformatik*, **33** (4), 289-299 (1991).

[33] T. Mochel, A. Oberweis, and V. Sänger. INCOME/STAR: The Petri net simulation concepts. *Systems Analysis - Modelling - Simulation, Journal of Modelling and Simulation in Systems Analysis*, **13**, 21-36 (1993).

[34] A. Oberweis. Workflow management in software engineering projects. *Proc. 2nd International Conf. on Concurrent Engineering and Electronic Design Automation*, Bournemouth, UK, 55-60 (1994).

[35] A. Oberweis and V. Sänger. Graphical query language for simulation runs. *Journal of Microcomputer Applications* (to appear 1994).

[36] A. Oberweis, V. Sänger, and W. Weitz. GAPS - A multiuser tool for graphical simulation of Petri nets. *Proc. 1st Joint Conf. of International Simulation Societies*, Zurich, Switzerland, 377-381 (1994).

[37] A. Oberweis, P. Sander, and W. Stucky. Petri net based modelling of procedures in complex object database applications. *Proc. IEEE 17th Annual International Computer Software and Applications Conf.*, Phoenix, Arizona, 138-144 (1993).

[38] A. Oberweis, T. Wendel, and W. Stucky. Teamwork coordination in a distributed software development environment. in *Innovationen bei Rechen- und Kommunikationssystemen*, Wolfinger, B. (Ed.), Springer, 423-429 (1994).

[39] N. Prakash, C. Rolland, and B. Pernici (Eds.). *Information System Development Process*. North-Holland (1993).

[40] O. Rauh and E. Stickel. Entity tree clustering - a method for simplifying ER design. *Proc. 11th International Conf. on the Entity-Relationship Approach*, Karlsruhe, Germany, 62-78 (1992).

[41] W. Reisig. Petri Nets. *EATCS Monographs on Theoretical Computer Science*, Springer (1985).

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

[43] H.-J. Schek and M. Scholl. The relational model with relation-valued attributes. *Information Systems*, **11** (2), 137-147 (1986).

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

[45] A. Sinha. Client-Server computing. *Communications of the ACM*, **35** (7), 77-98 (1992).

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

[47] A. Sølvberg and D.C. Kung. *Information Systems Engineering - an Introduction*. Springer (1993).

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

[49] B.C. Warboys (Ed.). *Software Process Technology*. Springer (1994).