INCOME/STAR: PROCESS MODEL SUPPORT FOR THE DEVELOPMENT OF INFORMATION SYSTEMS*

Wolffried Stucky, Andreas Oberweis, Gabriele Scherrer
Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe (TH)
D-W-7500 Karlsruhe 1
Germany

E-Mail: {stucky|oberweis|gse}@aifb.uni-karlsruhe.de

Abstract

INCOME/STAR is a conception of an integrated environment for the development and maintenance of large, distributed information systems. It extends INCOME, an already existing and commercially available tool for conceptual modelling and prototyping of information systems, by the following features:

- design of data and transactions distribution,
- physical design,
- hypertext interface for design dictionaries,
- cooperative system development,
- maintenance of large information systems,
- project management,
- support of different process model standards.

In this paper, the architecture of INCOME/STAR is introduced and the main features are surveyed. The paper focuses on one central aspect of INCOME/STAR, the process model support for the development of large information systems.

1 Introduction

An increasing demand for software support in various application fields such as industrial automation, business and administration requires individually tailored software solutions and efficient database supported information systems.

Information systems are not only supposed to be specially tuned 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.

* This work is partially supported by the Deutsche Forschungsgemeinschaft DFG under grant Stu 98/9-1 in the program "Verteilte DV-Systeme in der Betriebswirtschaft“.

Such requirements call for distributed data management, and information systems have to be conceived as *open* systems, which are characterized by the following features [Tsi89]:

- *Geography*: Systems are integrated in networks; future extensions of these networks are not known a priori.
- *Environment:* Systems are embedded in a heterogeneous software and hardware configuration which is subject to change.
- *Evolution:* System requirements are likely to change during operation (and even during development).

This paper deals with tool support for the development and maintenance of such open systems. Several research prototypes for information system development tools have been proposed (see e.g. [ADD85, JaD89, PBF89, SNH87, SSB90]) which mostly support conceptual (implementation independent) and logical data design, sometimes supported by prototyping facilities. But these tools lack a satisfying support for data and transactions distribution, physical data design, implementation and maintenance. Questions of selecting e.g. appropriate hardware or assessing database management systems for a specific task are usually not considered.
Commercially available software development tools [Bal91] also still have some deficiencies: Some of these tools completely ignore the central aspect of data modelling. If however data modelling is supported, then there usually does not exist a methodological integration of static and dynamic system aspects [ScN92].
Several existing database management systems already provide facilities for distributed data management; for most other systems such facilities are announced for the next future. There exists a lot of research work in the area of distributing data and transactions in computer networks [ÖzV91], but none of these concepts has been sufficiently integrated into an existing tool yet, which supports the whole information system life cycle. Database system developers can only rely on the database vendors' proprietary administration tools (if such tools already exist).

In the following we describe INCOME/STAR, a conception of an integrated environment for the development and maintenance of large distributed information systems. It extends INCOME, an already existing and commercially available tool for conceptual modelling and prototyping of information systems, by the following features:

- design of data and transactions distribution,
- physical design,
- hypertext interface for design dictionaries,
- cooperative system development,
- maintenance of large information systems,
- project management,
- support of different software development process model standards.

This paper focuses on the process model support provided by INCOME/STAR.

The paper is structured as follows: In Section 2 the use of process models as a generic framework for the system development process will be discussed briefly. Section 3 summarizes the main features of INCOME. Section 4 surveys the functionality of INCOME/STAR with respect to the provided process model support, while Section 5 discusses architectural aspects of INCOME/STAR. Section 6 describes future extensions and surveys some open problems.

2 Process Model Support for the Development of Large Information Systems

The development and maintenance of large information systems is a complex task. *Process models* provide a generic framework for the organization of the whole life cycle of an information system. There exist several gross types of process models (e.g. *waterfall model*, *evolutionary system development*, *spiral model* [Agr86, BoB88]) which have been refined to complete guidelines for software system development. Examples are *IEEE Standard*, *U.S. Military Standard*, *NASA Standard*, *ESA Standard* (for a survey of these standards see [DoT90]), *V-Modell* [BW91] or *ISOTEC* [ISO91].
Software customers and software suppliers must choose an appropriate process model before starting a software project. Usually software suppliers prefer a certain process model, which is possibly an in-house development. Sometimes software customers – especially in public administration – require a certain public standard process model for their software development projects.
The selected process model must usually be *tailored* (adjusted) to the specific needs of a specific software project. The tailoring process is done in cooperation between system customer and system supplier. Each tailoring decision should be documented, e.g. in the project plan. Some process models (for example the V-Modell [BW91]) include certain rules or plausibility checks for tailoring.

The use of a process model in a software project has several advantages, e.g.:
- The tailored process model can serve as the basis for agreement between customer and supplier side on how to organize the software project.
- The development efforts can be reduced.
- A reliable basis for estimating costs is provided.
- Quality of the project results is enhanced.

On the other hand the use of a process model in a software project causes a lot of additional work:

Since a complete process model is usually a rather complex document (consisting of possibly hundreds of pages), it requires a lot of teaching and training efforts for persons involved in a software project, to use the model efficiently. These efforts must also include the customer side.
During the whole project additional control effort is needed to check activities and documents with respect to the requirements of the process model.

An efficient application of a process model in practice requires tool support for at least the following aspects:

- tailoring,
- management of documents,
- monitoring of development activities,
- query facility for information about project status and project responsibilities,
- activity and capacity planning,
- exception handling,
- integrated hypertext user interface,
- coupling to tools for specific methods (e.g. editors, simulators, generators).

In Section 4 these aspects are discussed in more detail with respect to INCOME/STAR, an extension to INCOME, whose main features are surveyed in the following section.

3 INCOME

3.1 History

The concepts for INCOME (**I**nteractive **N**et-based **CO**nceptual **M**odelling **E**nvironment) were originally developed at the University of Karlsruhe between 1985 and 1990 [LNO89][1]. The main features of the university version of INCOME are: Integration of structural and behavioural system aspects (modelled in a *semantic data model* and *high level Petri nets)*, prototyping facilities and design dictionary support.
Based on these concepts, a commercially available methods and tools package was developed. The commercial product INCOME[2] [INC92, ScO92] is embedded in the ORACLE*CASE[3] [ORA92] product family. ORACLE*CASE supports CASE*Method, an information engineering approach for database oriented system development. INCOME extends the

[1] The project was supported by the Deutsche Forschungsgemeinschaft DFG under grant Stu 98/6 in the program "Interaktive betriebswirtschaftliche Informations- und Steuerungssysteme".
[2] Product of PROMATIS Informatik, Karlsbad
[3] Product of ORACLE Corp., Belmont, CA, USA

ORACLE*CASE environment by providing modelling facilities for behavioural aspects of the target system, such as procedure modelling, exception handling and temporal restrictions.

3.2 Basic Concepts and Components[4]

- **Petri Net Based Description of Behavioural System Aspects**

The INCOME method for modelling system behaviour bases on the formal concept of *predicate/transition nets* (Pr/T nets) [Gen87] – a high-level extension of Petri nets – which are used as a uniform description formalism for all relevant behavioural system aspects. The structure of the INCOME behaviour scheme is shown in *Figure 1*:

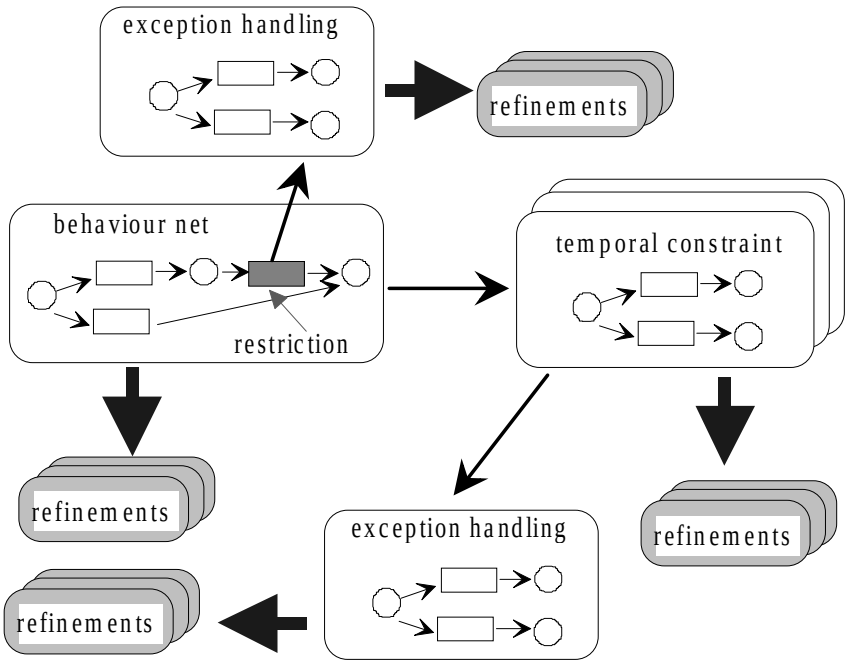

Figure 1: Structure of the INCOME Behaviour Scheme [INC92]

A system behaviour scheme consists of a set of behaviour nets connected through pre-defined relationships. The central component of a behaviour scheme is a hierarchy of behaviour nets (i.e. a top-level behaviour net and a set of refinements). Behaviour nets may contain static, transitional and temporal constraints, each requiring exception handling mechanisms. Constraints and exception handling mechanisms themselves are modelled by separate Pr/T nets or Pr/T net hierarchies.

[4] For a complete description of INCOME's functionality see the INCOME manuals [INC92].

Pr/T nets are suitable for visualizing and monitoring complex system activities and the relationships between activities. They allow a stepwise formalisation of system behaviour specifications as well as a stepwise refinement of system activities. Furthermore, Pr/T nets are directly executable and can be used for simulative validation of system designs.

- **Central Repository**

The central component of INCOME is a *repository* (*INCOME/Dictionary*), where all design documents and their interrelationships are stored in a pre-defined data base structure. Integrity of data is supported by a set of pre-defined integrity rules, and system developers are able to keep control over possible effects of design-decisions at any time.
Access to data is provided through special forms offering facilities for the management of

- information system procedures,
- static, transitional and temporal constraints and
- exception handling mechanisms.

In addition to this, data can be viewed and manipulated with different graphical editors (see next paragraph).
INCOME/Dictionary is integrated into ORACLE's CASE*Dictionary, where information about static aspects of data and function structures are stored.

- **Graphical User-Interface**

A Petri net editor (*INCOME/Designer*) serves as graphical front-end for retrieval and modification of information stored in the INCOME/Dictionary. Editing is done interactively with windowing and menu techniques. Presenting complex system behaviour schemes in a well-structured manner is managed through support of refinement and coarsening of Petri nets.
To ensure consistency of dictionary and graphics in a client-server environment a locking mechanism is provided. This locking mechanism is based on logical objects (like Petri net, entity/relationship diagram) and not on physical objects like row in a table or whole table.
An interface to ORACLE's CASE*Designer allows visualizing and editing object and function structures with an entity/relationship diagrammer, a cross-reference matrix editor, a function hierarchy diagrammer and a data flow diagrammer.

- **Prototyping and Automated Code Generation**

A transformation component *(INCOME/Generator*) supports transformation of requirement models into module design and furthermore into program code. In a first step, INCOME/ Generator uses Pr/T nets, entity/relationship diagrams and cross-reference matrices (relating e.g. entities and tables to each other) documented in CASE*Dictionary and INCOME/Dictionary to produce a module design which is then stored in CASE*Dictionary. In the next step C-code – extended with embedded SQL-calls – is derived from the module design.

As there is only a restricted amount of effort needed for program generation, INCOME/Generator may also be used for rapid prototyping.

4 Process Model Support in INCOME/STAR

The goal of the INCOME/STAR project is to provide additional support in INCOME for the following system development tasks:

a) Embedding newly developed information systems in an already existing organizational and technical environment.
b) Maintenance of existing information systems and adjustment to changes in the environment.
c) Developing federations of information systems.

This goal requires several extensions to INCOME. The management of large and complex software projects requires well coordinated development activities, efficient use and distribution of resources and control over the latest project status at any time. Provided that an efficient computer support is given, software development process model usage can be a powerful help to fulfil these tasks at any stage of the system life cycle.
The INCOME/STAR approach provides an elaborated process model support by describing the chosen process model in a formal notation and including it into the design dictionary. Coupled to various tools which are integrated under a unique hypertext user interface, initiation and monitoring of development activities and creation and administration of design documents is controlled by the process model and its requirements.
The following section describes the process model support provided by INCOME/STAR in detail.

4.1 Formal Description of Process Models

The formal description of the used process model is a prerequisite for computer support in the software development process. There exist several proposals where process model activities are described in a programming language or logic oriented style [BoB88,KaF88,Ost88,Wil88].
We think that it is more appropriate to describe the process model in the same notation as the software systems to develop: for the description of process model activities we use Petri nets. Some process model activities may occur sequentially, others in parallel (concurrently) or alternatively. These relationships can be adequately expressed in Petri nets. Stepwise refinement of activities leads to net hierarchies. The bottom level nets in a net hierarchy provide a precise description of process model activities whereas top level nets provide a gross overview about the process model. Manual and unstructured activities (like unstructured

communication) can be expressed by informal types of Petri nets, so-called *channel/agency nets*, where the net components are inscribed with natural language expressions.
Document structures (requirement documents, design documents, project management documents, ...) which are relevant in a process model, can be described by semantic data models (e.g. the entity/relationship model or the semantic hierarchy data model[5]).
Additional requirements with respect to the relationships between different documents or between documents and activities can be declaratively modelled by so-called *fact transitions* and *excluded transitions* [Obe92]. Fact transitions restrict the set of regular project states by rules like e.g. "If document d1 exists then document d3 does not exist" or "If document d2 contains chapter 3.2 then document d4 must contain chapter 3.2 too".
Excluded transitions restrict the set of regular project activity sequences by rules like e.g. "Activity A4 must not occur if activity A3 has already occurred" or "Activity A5 must not take more than 2 weeks".

4.2 Tailoring

Project specific tailoring means adapting a process model to the requirements of a certain project by omitting or adding certain activities or documents. Moreover, some of the documents required by the process model may be replaced by already existing ones. If so, it is necessary to record – together with information about possible deviations – that a certain document corresponds to another document in the process model.
In any case, tailoring must follow certain rules with respect to the interrelationships between activities and documents. If documents are to be created in addition to those required in the original process model, a description of their structure and cross references to other documents must be specified. For supplementary activities it is necessary to identify input and output documents as well as links to other activities and their role within the process schedule.
As described above, the INCOME/STAR approach models documents as a *data scheme* specified in a semantic data model and the software development process itself as a hierarchy of Petri nets. Therefore, tailoring will result in both changes of the net structures and changes of the document schemes. Editors with automated plausibility checking facilities based on pre-defined tailoring rules can be used to support these changes. The tailoring rules are again specified as fact transitions and excluded transitions (see Section 4.1) in special *tailoring nets*.

[5] The University version of INCOME supports the semantic hierarchy data model. Due to the integration into ORACLE*CASE, the commercial version of INCOME supports the entity/relationship model. In INCOME/STAR it is planned to support both types of semantic data models.

4.3 Creation and Management of Documents

An efficient creation and management system for documents must be supported by appropriate tools:
Document creation can be aided by editors which are able to check a document with respect to a required document structure. Provided that sufficiently fine locking mechanisms are supported, multiuser control management allows different users to manipulate different parts of one document at the same time.
Another important task is to control the way documents are altered, i.e. to make sure that users can always work on the most recent version of a document, and to administer older versions automatically. The necessary technical and administrative support is provided by a central dictionary for design documents with facilities for version control, multiuser control and recovery procedures.

4.4 Monitoring of Development Activities

Initiation and Coordination of Development Activities
Starting with documents available at the beginning of the project, initiation and coordination of development activities will be done computer-aided by monitoring requirements specified in the process model in terms of temporal constraints or cross-references between activities. Coordination of activities includes giving a survey of the current project status and dependencies among activities (see next paragraph), monitoring and adjusting project plans and assisting at choosing appropriate tools for the next activities.

Query Facility for Information about Project Status and Project Responsibilities
A project information query component enables every person involved in the project to get detailed information on the current project status and the corresponding activities specified in the process model whenever this is desired. The current project status is determined by the state of documents and activities: at a given time, documents may be completed, under creation or not yet created; activities are done, started or not yet initiated.
In addition to project status information, project responsibility information, e.g. name or phone number of the person responsible for a certain activity, can be queried.
The main task of the information facility described above is to free development personnel from time-consuming activities like looking up in process model manuals and keeping project plans manually up-to-date.

4.5 Resource and Capacity Planning

Simulation of Development Alternatives
With respect to possible activity sequences specified in the formal process model, different alternatives of subsequent activities at a given project state can be compared, e.g. by simulating different resource and capacity plans or time schedules.
Software projects are not static: adjustments or altering of activity sets may be initiated by the customer side if system requirements change during development or if changes in the hardware or software environment occur. On the supplier side, unforeseen events may require *exception handling* (see next paragraph). Simulation can help to predict the additional effort caused by user- or supplier-initiated changes and to adjust time and cost schedules.

Exception Handling
A major reason to provide alternative development activities in a project plan is the possible occurrence of irregular or unexpected events *(exceptions).* Generally spoken, exceptions in a software development project are situations where the requirements of the project specific process model or additional temporal constraints cannot be met, due to e.g. fluctuation or illness of development staff, unexpected technical failures or problems with subcontractors.
In order to guarantee that project deadlines can be kept, sufficient mechanisms for exception handling have to be established.
Most exceptions are caused by external events, i.e. events that are beyond control of the system developers. In contrast to the occurrence of integrity violations in database systems, it is usually not possible to correct such error situations by simply rolling back activities (backward recovery).
Instead, the main goal of exception handling mechanisms should be to keep the effect of an error situation as limited as possible. One possible mechanism could be to provide an alternative set of activities to be executed if an unforeseen shortage of resources occurs. It may be possible, for example, to meet the time schedule of a retarded project by a higher degree of parallel execution or an acceleration of certain activities, e.g. by providing additional personnel.
An effective exception handling strategy should organize the distribution of limited resources among activities with respect to their importance within the project. Therefore, an *ex ante* identification of critical activities is required.

4.6 Hypertext User Interface

An important feature of process model support in INCOME/STAR is the *hypertext user interface*. Hypertext [Con87, Nie90] is a novel approach for the management of data, which is especially appropriate if the data is (partially) informal and unstructured. The stored data (consisting of text, pictures, diagrams, tables, etc.) is divided into certain data units which

may be arbitrary *linked* to each other in a so-called *hypertext network*. A data unit (so-called *node* in the hypertext network) may be e.g. a certain requirements document, the project activity schedule, a chapter in the system documentation. Links may exist e.g. from a certain design document to the related requirement documents. The user of a database with a hypertext interface *navigates* through the underlying hypertext network, without having to learn a specific query language. Hence hypertext is especially appropriate for unskilled users.
Hypertext support for software engineering has already been proposed e.g. in [CyR92, GaS87], but mostly in a rather restricted application area.
The planned hypertext support in INCOME/STAR allows navigational access to the process model documentation as well as to the documents which are created during the development process. For each document type a certain node type is provided. There exist different types of links, each having a specific semantics and each having a different graphical representation. Examples are: *explaining links* which connect formal documents with informal explaining documents; *refining links* which connect an activity or a data object with its refining activities or data objects; *tool invocation links* which connect documents to certain tools; *responsibility links* which relate certain persons to certain objects or activities; *comment links*, which connect personal comments to a certain document; ...
All phases of the information system life can be supported by the *tutoring*, *(co-)authoring* and *documentation* capabilities of hypertext. A detailed survey about the possibilities is given in [NeO92].

4.7 Tools Interface

The choice of a process model does not necessarily fix methods and tools – e.g. for data modelling, prototyping, database or module design – that are used to carry out a specific project. Therefore, INCOME/STAR is designed to be coupled to various tools for specific methods. The coupling is provided via the design dictionary interface, which allows access to all design documents.
Information about methods and tools to use is contained in the project plan. Invocation of tools is possible via the hypertext user interface: If a certain node, e.g. containing a Petri net, is activated, then the respective Petri net editor or a Petri net simulator may be invocated. A node, containing source-code may be linked to a text editor and a compiler.

5 Architecture of INCOME/STAR

Figure 2 gives a gross overview about the architecture of INCOME/STAR.

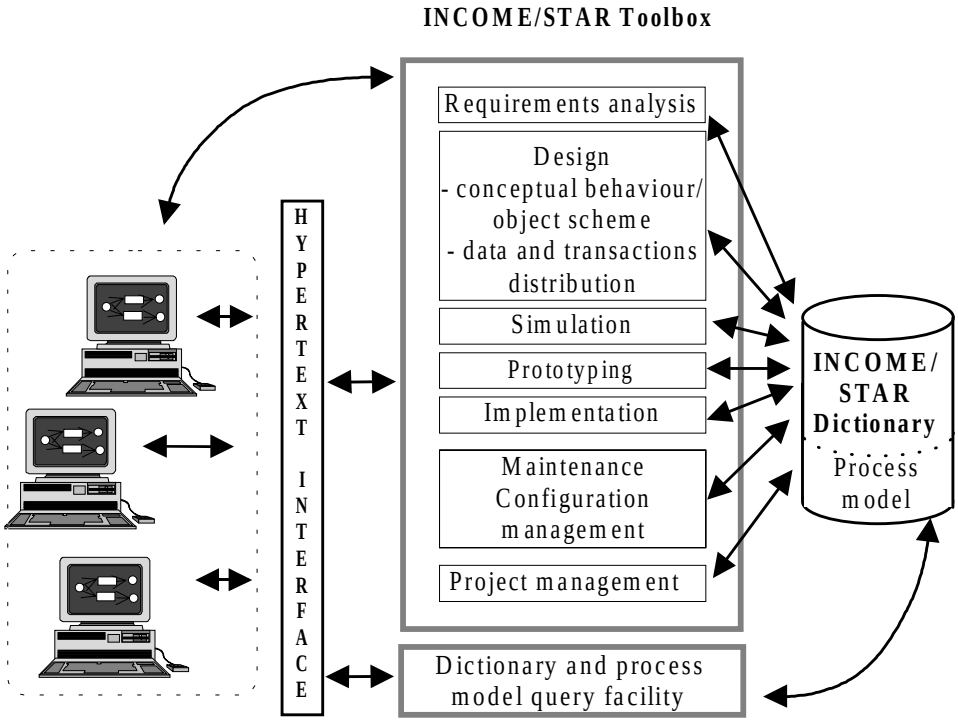

Figure 2: Architecture of INCOME/STAR

As in the predecessor system, INCOME, a central repository – *INCOME/STAR Dictionary* – is the key component for every development and maintenance task as it enables system developers to keep control over various documents.
As an important new feature, the INCOME/STAR dictionary does not only contain the design documents required by the chosen process model but also information about the process model itself and the way process model objects (documents, activities, constraints, exception handling mechanisms, ...) are related to each other. This is essential to provide process model support for different process models as efficiently as postulated in Section 4.
A hypertext user interface to the repository supports access to the design data and maintenance of design documents and serves as query facility for project status and project responsibility information (see Section 4.4).
Various tools are coupled to the repository and may either be accessed via the hypertext interface or be called directly.

The main features supported by the toolbox are:

• *Analysis and Formal Description of Requirements*

Aided by a special editor, informal information about system requirements can be transformed into a structured form of pre-defined object, activity and event glossaries (cf. [ADD85]). The editor offers automated structure, completeness and consistency checking facilities (e.g. identifying possible synonyms).

• *Conceptual Modelling of Object Structures and System Behaviour*

Graphical representations of object structures (entity/relationship schemes, semantic hierarchy data schemes) and system behaviour (Petri nets) can be generated interactively aided by graphical editors.

The functionality of these tools is not limited to the capability of a simple drawing program: based on the information in the design dictionary, graphical representations can be generated automatically and then – if desired – interactively tuned. Hierarchical organization of documents (e.g. refinements and coarsening of Petri nets) is supported as well as automated structure checking.

• *Design of Data and Transactions Distribution*

The design of distributed systems is supported by generators using certain analytical methods (cf. [ÖzV91]), e.g. to support segmentation and allocation of data in distributed database applications. It is planned to provide facilities for the tuning of existing distributed database applications with respect to the enhancement of reliability or performance.

• *Simulation and Prototyping*

Simulation means interpreting design documents to validate

– the completeness and correctness of the design documents for the target system and components,
– user interfaces,
– interfaces between components,
– system behaviour in exceptional situations,
– reachability of certain system states,
– temporal system behaviour.

In the first two cases, the main objective of simulation is to improve communication between system developers and users by giving the user the possibility to get familiar with the system's user interface and functionality. This is also denoted as prototyping.

INCOME/STAR supports simulation and prototyping in the above sense. The chosen specification language – Petri nets – can be directly used for simulation. As INCOME/STAR is especially conceived to support the design and maintenance of distributed systems, distributed simulation and prototyping is supported, i.e. several simulators – each running on a workstation in a computer network and each representing different system components or groups of components – are coupled together.

To support embedding of newly developed information systems in an already existing environment, simulators are coupled to external hardware or software.
A detailed description of INCOME/STAR's simulation concept is given in [MOS93].

• *Implementation*
Generators transform the dictionary information (entity/relationship schemes, function hierarchies, Petri net structures) into a structured module design (module definition, module hierarchies, assignment of tables and rows to a module). In a next step, database generators derive forms, reports and menus from the module design, while program generators use module hierarchies and Petri net information to generate C-code and embedded SQL.
As an extension to INCOME, a PEARL-code generator for real-time control system support is planned.

• *Maintenance and Configuration Management*
Maintenance of existing information systems includes
– changes in the object structure (add or remove entity types, relation schemes or attribute types),
– changes of behavioural aspects (add or remove restrictions, temporal constraints, transitions, ...),
– adding functionality (provide additional views, reports, user categories, screens, ...),
– adaptation to changes in the hardware or software environment and addition of interfaces to new components.

Most process models specify maintenance and quality assurance activities (see e.g. [BW91]). To provide an appropriate tool support, coordinated use of all tools described earlier is needed: carrying out a maintenance task requires structure and completeness checking facilities to manage changes within the same document, and hypertext-aided navigation through all other documents affected by the change. Simulation and prototyping tools can be used to examine possible effects of planned changes on the system's behaviour.
To keep different versions created by different persons under control, INCOME/STAR Dictionary is provided with facilities for configuration management, multiuser control and recovery procedures.

As mentioned before, INCOME/STAR is conceived to support different process models. Therefore it is open for the integration of other methods and tools, e.g. for project management tasks.

6 Summary and Outlook

In this paper we have described concepts for process model support which are currently integrated into INCOME/STAR, an environment for the development and maintenance of distributed information systems. A gross overview about the architecture of INCOME/STAR was given.

INCOME/STAR will support different process models. An open question still remains with respect to the selection of a specific process model for a certain project. How can the software supplier or the software customer select the process model which is most appropriate for the respective project? Another question in this context is how to measure the quality of project work with respect to a given process model.
Furthermore, there is still some methodological work necessary in the area of Petri nets with respect to the goal of INCOME/STAR. To examine aspects like reliability in distributed environments it is necessary to consider so-called *stochastic Petri nets.* To model complex objects (e.g. office documents) with possibly set-valued attributes it is necessary to use an appropriate type of high-level Petri nets, which provides an inscription language to select and manipulate these objects. A first step into this direction is described in [OSS92]. An open issue still is the integration of high-level Petri nets and stochastic Petri nets.
A more implementation oriented aspect of our future work is the support of parallel languages. Up to now, programs can be generated from a given Petri net specification, where these programs may run in parallel at different sites in a computer network. Currently no support is provided for the generation of programs in parallel languages like Parallel C.

References

[ADD85] A. Albano, V. DeAntonellis, A. DiLeva (Eds.): Computer-Aided Database Design, North-Holland Publ. Comp, 1985

[Agr86] W.W. Agresti (Ed.): New Paradigms for Software Development, IEEE Computer Society Press, 1986

[Bal91] H. Balzert: CASE Systeme und Werkzeuge, Reihe Angewandte Informatik, Bd. 7, BI.-Wiss.-Verlag, 2. Edition, 1991 (in German)

[BoB88] B. Boehm, F. Belz: Applying process programming to the spiral model, in: C. Tully (Ed.): Proc. 4th Int. Software Process Workshop, Moretonhampstead, 1988, p. 46-56

[BW91] Software-Entwicklungsstandard der Bundeswehr, "Vorgehensmodell", Allgemeiner Umdruck, Bundesamt für Wehrtechnik und Beschaffung, Koblenz, February 1991 (in German)

[Con87] Conklin, J.: Hypertext: An introduction and survey. Survey and tutorial series. *IEEE Computer*, September 1987, p. 17-41

[CyR92] J.L. Cybulski, K. Reed: A hypertext based software engineering environment,: *IEEE Software*, March 1992, p. 62-68

[DoT90] M. Dorfman, R.H. Thayer (Eds.): Standards, Guidelines and Examples on System and Software Requirements Engineering, IEEE Computer Society Press, 1990

[GaS87] P. Garg, W. Scacchi: On designing intelligent hypertext systems for information management in software engineering, in: Proc. Hypertext 87, 1987, p. 409-432

[Gen87] H.-J. Genrich: Predicate/transition nets, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.): Advances in Petri nets 86, Vol. I, Springer-Verlag, 1987, p. 207-247

[INC92] INCOME-Benutzerhandbücher: INCOME/Designer, INCOME/Dictionary, INCOME/Generator, PROMATIS Informatik, Karlsbad, 1992

[ISO91] ISOTEC-Handbücher: Vorgehenskonzept, Administrationskonzept, Projekt-managementkonzept, Qualitätssicherung, PLOENZKE Gruppe, Kiedrich, June 1991 (in German)

[JaD89] M. Jarke, DAIDA Team: DAIDA: Konzeptuelle Modellierung und wissensbasierte Unterstützung von Softwareprozessen, in: W. Brauer, C. Freska (Eds.): Proc. GI-Kongreß Wissensbasierte Systeme, Springer-Verlag, 1989, p. 440-452 (in German)

[KaF88] G.E. Kaiser, P.H. Feiler: An architecture for intelligent assistance in software development, in: Proc. 9th IEEE Int. Conference on Software Engineering, Monterey, 1988, p. 180-188

[LNO89] G. Lausen, T. Németh, A. Oberweis, F. Schönthaler, W. Stucky: The INCOME approach for conceptual modelling and rapid prototyping of information systems, in: Proc. CASE89. The First Nordic Conference on Advanced Systems Engineering, Stockholm, 1989

[MOS93] Th. Mochel, A. Oberweis, V. Sänger: INCOME/STAR: The Petri net simulation concepts, *Systems Analysis - Modelling - Simulation, Journal of Mathematical Modelling and Simulation in Systems Analysis* (to appear 1993)

[NeO92] S. Neubert, A. Oberweis: Einsatzmöglichkeiten von Hypertext beim Software Engineering und Knowledge Engineering, in: R. Cordes, N. Streitz (Eds.): Proc. Hypertext und Hypermedia 92, Munich, Informatik aktuell, Springer-Verlag, 1992, p. 162-174 (in German)

[Nie90] J. Nielsen: Hypertext & Hypermedia, Academic Press, San Diego, 1990

[Obe92] A. Oberweis: Spezifikation von Mechanismen zur Ausnahmebehandlung mit Petri-Netzen, *at -Automatisierungstechnik* 40, 1, 1992, p. 21-30 (in German)

[ÖzV91] M.T. Özsu, P. Valduriez: Principles of Distributed Database Systems, Prentice-Hall, 1991

[ORA92] ORACLE*CASE User's Guide and Reference**:** CASE*Dictionary, CASE*Designer, CASE*Generator, ORACLE Corp., Redwood City, 1992

[OSS92] A. Oberweis, P. Sander, W. Stucky: Modellierung von Abläufen in NF2-Datenbanken durch höhere Petri-Netze, in: R. Studer (Ed.): Proc. 2. Workshop Informationssysteme und Künstliche Intelligenz, Ulm, Informatik-Fachberichte 303, Springer-Verlag, 1992, p. 95-112 (in German)

[Ost88] L. Osterweil: Software processes are software too, in: Proc. 9th IEEE Int. Conference on Software Engineering, p. 2-13, Monterey, 1988

[PBF89] B. Pernici, F. Barbic, M.G. Fugini, R. Maiocchi, J.R. Rames, C. Rolland: C-TODOS: An automatic tool for office system conceptual design, *ACM Transactions on Information Systems*, Vol. 7, No. 4, 1989, p. 378-419

[ScN92] F. Schönthaler, T. Németh: Software-Entwicklungswerkzeuge: Methodische Grundlagen, B.G. Teubner Verlag, 2. Auflage, 1992 (in German)

[ScO92] F. Schönthaler, A. Oberweis: Entwicklung datenbankgestützter Automatisierungssysteme mit INCOME und ORACLE*CASE, in: Proc. 2. Fachtagung Entwurf komplexer Automatisierungssysteme, Braunschweig, May 1992 (in German)

[SNH87] G. Saake, L. Neugebauer, U. Hohenstein, H.-D. Ehrich: Konzepte und Werkzeuge für eine Datenbankentwurfsumgebung, Technische Universität Braunschweig, Informatik-Bericht Nr. 87-05, 1987 (in German)

[SSB90] B. Steinholtz, A. Solvberg, L. Bergman: Advanced Information Systems Engineering, LNCS 436, Springer-Verlag, 1990

[Tsi89] D. Tsichritzis Object-oriented development for open systems, in: D. Tsichritzis, (Ed.): Object Oriented Development, p. 1-13, Centre Universitaire D'Informatique, Université de Genève, Genève, 1989

[Wil88] L.G. Williams: Software process modelling: A behavioural approach, in: Proc. 10th IEEE Int. Conf. on Software Engineering, Singapore, 1988, p. 174-186