INCOME/STAR-ProMISE: Process-centered Information System Development

Gabriele Scherrer
Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe (TH)
D-76128 Karlsruhe/Germany

E-Mail: scherrer@aifb.uni-karlsruhe.de

1 Process-centered Information System Development

In *process-centered development environments*, the software development process is automatically guided and supported by a process model. In this context, a process model can be defined as a formal, executable description of the development process in terms of *activities* and *results (documents*).
Traditionally, process models - like the well-known waterfall-model and its variants - reflect the *software life cycle* with its different stages: software projects are broken down into a sequence of successive stages. The results of one stage serve as input for the following stage. Each stage is connected to the preceding one by a feedback loop.
Criticism of stagewise process models mainly addressed the following aspects:
Lack of flexibility: Requirements change frequently due to market factors, new technologies or strategical decisions. Hence, software development is a highly dynamic process which should not be rigorously separated in stages.
Neglect of maintenance: In spite of the well-known fact that the maintenance phase normally is the longest and most expensive period in a software life cycle, stagewise process models regard it as a kind of appendix of the preceding development stages.
Lack of attention for human factors: Many problems in the software development field arise from communication problems between developers and future users of a system. This is a point where stagewise models have some of their most important drawbacks as they do not provide facilities for user integration.

This criticism led to some new software development paradigms proposed in the mid-eighties [Agr86]: Concepts like *evolutionary system development, prototyping, operational specification, transformational implementation* and *software reuse* influenced software engineering.
Yet, there is still an unresolved contradiction between two key requirements to process models: on the one hand, they should result in a structured, well-planned development process; on the other hand, flexibility is required. To bridge this gap, efforts were made to integrate different approaches.
In the meantime, new trends became apparent in the field of process modeling:
Distribution: Systems may be geographically distributed and are quite frequently integrated in networks. This means that process models must offer description facilities for a system's topology, i.e. new (or adaptive) result types are needed. Another consequence of distribution is that it suggests the use of *cooperative development techniques* [OWS94], which may influence the process model's activity types.
Interoperability: Systems are supposed to communicate and exchange data with other systems. Consequently, an open information infrastructure is needed.
Side activities: In addition to activities being directly coupled with system development, there are other activities involved in the future success of a software product:

product documentation, quality assurance or user training are highly important activities, which should be integrated into a process model [Chr92].

2 ProMISE - a Process Model for Information System Evolution

The above discussion has been taken into consideration when ProMISE - a process model for information system evolution - was developed [SOS94]. ProMISE is part of INCOME/STAR, a repository-based environment for the development of distributed information systems.

The development process with INCOME/STAR includes database and application program generation, distribution of data and processes and parallelization of processes. It is supported by prototyping based on the conceptual schema. High level Petri nets are used for the representation of the behaviour schema [OSS93] and a semantic object model for the representation of the data schema. The INCOME/STAR repository supports shared access to the design documents and by this supports concurrent software engineering.

2.1 Basic Features of ProMISE

Figure 1 gives a generic representation of a development stage in ProMISE. A specific design document ('result type') has got a certain status as, e.g., requirements scheme document, 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 grey level indicating the degree of formality of this activity. Documents belonging to different parts of the system may be handled in parallel by different persons, i.e. stages may be executed in parallel or overlap where it is possible. Sometimes a situation may require a go-back to an earlier stage for a certain document, e.g. if requirements are added or changed, i.e. the model is dynamically modified whenever the development process requires such dynamic changes.

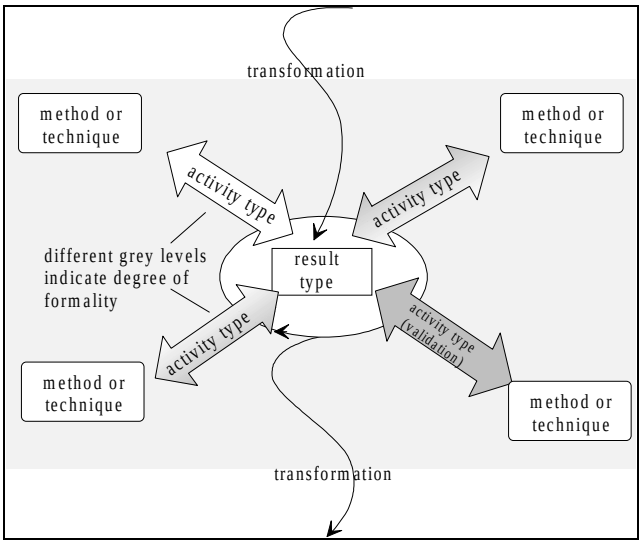

Fig. 1: Generic representation of a development stage in ProMISE

Usually, document creation starts with a - more or less formal - transformation step, converting resulting documents of the preceding stage into (initial) documents of the current phase. These documents are iteratively adjusted by a sequence of activities (refinement, structuring, modeling, information collection and quality-checking steps etc.).

At the end of each iteration, the existing documents will be examined in a validation step. Whenever it makes sense, users will be involved in this process. If a document's quality is acceptable, it may be transformed into an initial document of the successive stage. Otherwise, it has to be modified, which means a new iteration of information collection, modification and quality checking steps.
A specific description in the notation introduced in Figure 1 is available for *strategic planning, project specific planning, requirements collection and analysis, conceptual modeling, database design and implementation, program design and implementation* [SOS94], but not for maintenance. In fact, maintenance is not considered as a *phase* at all, but as a *process of evolutionary system enhancement*, resulting in a new iteration of the development cycle.

2.2 Computer Support

An efficient application of a process model in practice needs tool support for monitoring of development activities, workflow management, document management, control of project status and project responsibilities, activity and capacity planning and coupling to tools for specific methods.
This requires a refined formal description of the process model. In INCOME/STAR, the process model is specified as a hierarchy of Petri nets. This notation permits an adequate description of relationships between activities: *Activity sequences* can be modelled as well as *conflict situations* between activities (mutual exclusion) and *concurrency.* Stepwise refinement of activities leads to net hierarchies. 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. These top level nets provide a gross overview about the process model and may be used as a graphical notation for communication between different groups of people involved in a software project. The bottom level of the hierarchy provides a precise NR/T-Net-description of the process model which is directly executable by our Petri Net simulator.
A workflow manager [Obe94] supports planning and modeling of development activities based on the Petri net model. It monitores and controls the execution of workflows and supports resource allocation.

References

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

[Chr92] G. Chroust: *Modelle der Software-Entwicklung*, Oldenburg Verlag, München 1992

[Obe94] A. Oberweis: Workflow Management in Software Engineering Projects, in: Proc. of the 2nd International Conference on Concurrent Engineering and Electronic Design Automation, Bournemouth/UK, April 1994, p. 55-60

[OSS93] A. Oberweis, P. Sander, W. Stucky: Petri net based modelling of procedures in complex object database applications, in: D. Cooke (Ed.): Proc. IEEE 17th Annual International Computer Software and Applications Conference, Phoenix/Arizona, November 1993, p. 138-144

[OWS94] A. Oberweis, T. Wendel, W. Stucky: Teamwork coordination in a distributed software development environment, in: Proc. GI-Fachgepräch Communication and Coordination in Distributed Corporate Application Systems, IFIP-Workshop, Hamburg/Germany, August 1994 (to appear)

[SOS94] G. Scherrer, A. Oberweis, W. Stucky: ProMISE - a process model for information system evolution, in: Proc. 3rd Maghrebian Conference on Software Engineering and Artificial Intelligence, Rabat/Morocco, April 1994, p. 27-36