Systematic Development
of
Formal Software Process Models'

Wolfgang Deiters', Volker Gruhn', Wilhelm Schäfer'

'Software Technology Center
Helenenbergweg 19
D-4600 Dortmund 50
Federal Republic of Germany

'University of Dortmund
Computer Science
Software Technology
P.O. Box 500 500
D-4600 Dortmund 50
Federal Republic of Germany

Abstract

This paper proposes a structured approach to the incremental development of generic Models for Software Processes. Structured means that the developer is guided by an underlying systematic method when specifying the complex set of items constituting a complete description of a Software Process. Incremental means that it is possible to test, execute, and analyse incomplete specifications. Generic means that suitable mechanisms are provided to adapt a particular model to the specific requirements of a single company or user. The approach is based on a formal language which merges three existing approaches, namely a Data Definition Language, Function Nets as an extension of high-level Petri Nets, and graph replacement systems, into one homogeneous Software Process modeling language. It is explained how this language provides suitable means to specify the static features of a process model, the dynamics of processes, and the modifications of the processes. Finally, it is sketched how a process model is defined, i.e. this language can be executed for simulation reasons as well as for the control of a Software Process supported by the tools within the Software Development Environment.

Keywords:
Software Development Environment, Software Factory, Software Process, Model for Software Processes, method, high-level Petri Nets, Function Nets, graph replacement systems

*) The work described here is partially sponsored by the ESPRIT project ALF and the EUREKA project ESF.
1 Introduction and Related Work

Recent research efforts (e.g. [Lehm84], [BGS84], [Boeh86], [EP86], [WC84]) indicate that an explicit model for Software Processes (SPs) is regarded nowadays as one of the most essential but still missing parts of Software Development Environments (SDEs). As Osterweil states in [Oste87], it seems to be incredible that even though it is known that an adequate choice of a problem-adapted model for software development has a big impact on productivity and quality of produced software, explicit descriptions of the process are missing in many environments. Consequently, quite a number of research projects have recognized the necessity of including a process model into a SDE as a means to improve the industrial-like production of software (e.g. ESF [SW88], ALF [TA87], Arcadia [TBC088]).

ESF and ALF, for example, propose to apply methods known from knowledge engineering to model the knowledge about Software Processes. The Arcadia project suggests to describe environments as the union of a fixed and a variable part and to offer the opportunity to test alternative process descriptions. Those approaches ultimately result in an explicit description of Software Processes. The mentioned projects (and a number of others) introduced the term Software Factory to indicate that the support offered by a SDE is oriented towards a more industrial-like production of software.

The main research directions in process modeling focus on fixing the scope and required features of Process Modeling Languages [Kel188], [Sutt88], then on developing such languages [Robe88], [0088], [DABB88], [Hitc88], [Kais88], [Wil188], [ARSV88], on providing an environment architecture supporting the modeling of Software Processes [TBC088], [Dows87], and on investigating needed basic supporting facilities for such environments like an Object Management System or a specific tool set [Oste88].

Some common intrinsic features of process modeling have already evolved from this early work. First, there should be a clear distinction between a model and its instantiation. A model describes the relevant software development items and their relation to each other. A model is instantiated when values are assigned to those items, i.e. a certain object of a specific type is created. Thus, a model definition can be considered as a type definition for a class of similar SPs (e.g. SPs which all have the same underlying Software Life Model and the same Project Management Model), whereas a concrete SP is an instantiation of the corresponding type. Secondly, the instantiation procedure should enable one to tailorize a particular SP-model towards the peculiarities of a particular company, i.e. 'the SP model must essentially be some kind of generic type definition. Peculiarities of a particular company include items like number of programmers, particular set of milestones, special treatment of certain documents (e.g. a written source code has always to be checked by the project